Documentación de Trident 25.10

Trident

NetApp
February 25, 2026

This PDF was generated from https://docs.netapp.com/es-es/trident-2510/index.html on February 25, 2026. Always check docs.netapp.com for the latest.

Tabla de contenidos

Documentación de Trident 25.10 . . . 1
Notas de publicación . . . 2
Novedades . . . 2
Novedades en 25.10 . . . 2
Cambios en 25.06.2 . . . 4
Cambios en 25.06.1 . . . 4
Cambios en 25.06 . . . 5
Cambios en 25.02.1 . . . 7
Cambios en 25.02 . . . 7
Cambios en 24.10.1 . . . 9
Cambios en 24.10 . . . 9
Cambios en 24.06 . . . 11
Cambios en 24.02 . . . 12
Cambios en 23.10 . . . 12
Cambios en 23.07.1 . . . 13
Cambios en 23.07 . . . 13
Cambios en 23.04 . . . 14
Cambios en 23.01.1 . . . 15
Cambios en 23.01 . . . 15
Cambios en 22.10 . . . 16
Cambios en 22.07 . . . 17
Cambios en 22.04 . . . 18
Cambios en 22.01.1 . . . 19
Cambios en 22.01.0 . . . 19
Cambios en 21.10.1 . . . 20
Cambios en 21.10.0 . . . 20
Problemas conocidos . . . 21
Encuentra más información . . . 22
Versiones anteriores de la documentación . . . 22
NetApp soporte Trident para sistemas de almacenamiento ONTAP ASA r2 . . . 23
Operaciones compatibles . . . 23
Operaciones no admitidas . . . 23
Problemas conocidos . . . 24
Restaurar backups de Restic de archivos grandes puede fallar . . . 24
Empezar . . . 25
Conoce más sobre Trident . . . 25
Conoce más sobre Trident . . . 25
Arquitectura de Trident . . . 26
Conceptos . . . 29
Inicio rápido para Trident . . . 33
¿Qué sigue? . . . 34
Requisitos . . . 35
Información crítica sobre Trident . . . 35

Frontends compatibles (orquestadores) . . . 35
Backends compatibles (almacenamiento) . . . 36
Soporte de Trident para KubeVirt y OpenShift Virtualization . . . 36
Requisitos de funciones . . . 37
Sistemas operativos host probados . . . 37
Configuración del host . . . 37
Configuración del sistema de almacenamiento . . . 38
Puertos de Trident . . . 38
Imágenes de contenedores y versiones de Kubernetes correspondientes . . . 38
Instala Trident . . . 39
Instala usando el operador Trident . . . 39
Instala usando tridentctl . . . 39
Instala usando el operador certificado de OpenShift . . . 39
Usa Trident . . . 40
Prepara el nodo trabajador . . . 40
Seleccionar las herramientas adecuadas . . . 40
Descubrimiento de servicios de nodo . . . 40
Volúmenes NFS . . . 41
volúmenes iSCSI . . . 41
Volúmenes NVMe/TCP . . . 45
Volúmenes SCSI sobre FC . . . 46
Prepárate para aprovisionar volúmenes SMB . . . 49
Configura y gestiona backends . . . 50
Configura backends . . . 50
Azure NetApp Files . . . 51
Google Cloud NetApp Volumes . . . 70
Configura un backend de NetApp HCI o SolidFire . . . 87
Controladores SAN de ONTAP . . . 93
Controladores ONTAP NAS . . . 123
Amazon FSx for NetApp ONTAP . . . 160
Crea backends con kubectl . . . 194
Gestiona backends . . . 201
Crea y gestiona clases de almacenamiento . . . 211
Crear una clase de almacenamiento . . . 211
Administra clases de almacenamiento . . . 214
Aprovisiona y gestiona volúmenes . . . 216
Aprovisiona un volumen . . . 216
Ampliar volúmenes . . . 220
Importar volúmenes . . . 231
Personaliza los nombres y las etiquetas de los volúmenes . . . 241
Comparte un volumen NFS entre espacios de nombres . . . 244
Clonar volúmenes en distintos espacios de nombres . . . 248
Replica volúmenes usando SnapMirror . . . 251
Usa la topología CSI . . . 257
Trabaja con instantáneas . . . 265

Trabaja con instantáneas de grupo de volúmenes . . . 273
Administra y supervisa Trident . . . 278
Actualizar Trident . . . 278
Actualizar Trident . . . 278
Actualiza con el operador . . . 279
Actualiza con tridentctl . . . 284
Administra Trident usando tridentctl . . . 285
Comandos y banderas globales . . . 285
Opciones de comando y flags . . . 287
Compatibilidad con complementos . . . 293
Monitorea Trident . . . 293
Descripción general . . . 293
Paso 1: define un objetivo de Prometheus . . . 293
Paso 2: Crea un ServiceMonitor de Prometheus . . . 294
Paso 3: Consultar métricas de Trident con PromQL . . . 296
Conoce la telemetría de Trident AutoSupport . . . 297
Desactivar métricas de Trident . . . 298
Desinstala Trident . . . 298
Determina el método de instalación original . . . 298
Desinstala una instalación del operador Trident . . . 298
Desinstala una `tridentctl` instalación . . . 299
Trident para Docker . . . 300
Requisitos previos para el despliegue . . . 300
Verifica los requisitos . . . 300
Herramientas NVMe . . . 302
Herramientas FC . . . 303
Implementa Trident . . . 305
Método de plugin gestionado por Docker (versión 1.13/17.03 y posteriores) . . . 305
Método tradicional (versión 1.12 o anterior) . . . 307
Inicia Trident al arrancar el sistema . . . 309
Actualizar o desinstalar Trident . . . 309
Actualizar . . . 310
Desinstalar . . . 311
Trabaja con volúmenes . . . 311
Crear un volumen . . . 311
Eliminar un volumen . . . 312
Clonar un volumen . . . 312
Accede a volúmenes creados externamente . . . 314
Opciones de volumen específicas del controlador . . . 314
Recopilar registros . . . 319
Recopila registros para solucionar problemas . . . 319
Consejos generales para la resolución de problemas . . . 320
Gestiona varias instancias de Trident . . . 320
Pasos para el plugin gestionado por Docker (versión 1.13/17.03 o posterior) . . . 320
Pasos para el método tradicional (versión 1.12 o anterior) . . . 321

Opciones de configuración del almacenamiento . . . 321
Opciones de configuración global . . . 321
Configuración de ONTAP . . . 322
Configuración del software Element . . . 330
Problemas conocidos y limitaciones . . . 332
Al actualizar Trident Docker Volume Plugin a 20.10 y versiones posteriores desde versiones anteriores, la actualización falla con el error no such file or directory. . . . 332
Los nombres de los volúmenes deben tener al menos 2 caracteres de longitud. . . . 333
Docker Swarm tiene ciertos comportamientos que impiden que Trident lo admita con cada combinación de almacenamiento y controlador. . . . 333
Si se está aprovisionando un FlexGroup, ONTAP no aprovisiona un segundo FlexGroup si el segundo FlexGroup tiene uno o más agregados en común con el FlexGroup que se está aprovisionando. . . . 333
Buenas prácticas y recomendaciones . . . 334
Despliegue . . . 334
Implementa en un espacio de nombres dedicado . . . 334
Usa cuotas y límites de rango para controlar el consumo de almacenamiento . . . 334
Configuración de almacenamiento . . . 334
Descripción general de la plataforma . . . 334
Mejores prácticas de ONTAP y Cloud Volumes ONTAP . . . 334
SolidFire mejores prácticas . . . 339
¿Dónde encontrar más información? . . . 341
Integrar Trident . . . 341
Selección y despliegue de controladores . . . 342
Diseño de storage class . . . 344
Diseño de pool virtual . . . 345
Operaciones de volumen . . . 346
Servicio de métricas . . . 350
Protección de datos y recuperación de desastres . . . 351
Replicación y recuperación de Trident . . . 351
Replicación y recuperación de SVM . . . 351
Replicación y recuperación de volúmenes . . . 353
Protección de datos Snapshot . . . 353
Automatizando la conmutación por error de aplicaciones con estado con Trident . . . 353
Detalles sobre la desconexión forzada . . . 353
Detalles sobre la conmutación automática al respaldo . . . 354
Seguridad . . . 359
Seguridad . . . 359
Linux Unified Key Setup (LUKS) . . . 360
Cifrado Kerberos en vuelo . . . 367
Protege las aplicaciones con Trident Protect . . . 375
Conoce Trident Protect . . . 375
¿Qué sigue? . . . 375
Instala Trident Protect . . . 375
Requisitos de Trident Protect . . . 375
Instala y configura Trident Protect . . . 379

Instala el plugin CLI de Trident Protect . . . 383
Personaliza la instalación de Trident Protect . . . 387
Administra Trident Protect . . . 392
Administra la autorización y el control de acceso de Trident Protect . . . 392
Supervisa los recursos de Trident Protect . . . 399
Genera un paquete de soporte Trident Protect . . . 404
Actualizar Trident Protect . . . 406
Gestiona y protege las aplicaciones . . . 408
Usa los objetos de Trident Protect AppVault para gestionar buckets . . . 408
Define una aplicación para gestión con Trident Protect . . . 422
Protege las aplicaciones usando Trident Protect . . . 426
Restaurar aplicaciones . . . 438
Replica aplicaciones usando NetApp SnapMirror y Trident Protect . . . 456
Migra aplicaciones usando Trident Protect . . . 473
Administra los hooks de ejecución de Trident Protect . . . 477
Desinstala Trident Protect . . . 489
Blogs de Trident y Trident Protect . . . 490
Blogs de Trident . . . 490
Blogs de Trident Protect . . . 490
Conocimiento y soporte . . . 492
Preguntas frecuentes . . . 492
Preguntas generales . . . 492
Instala y usa Trident en un clúster de Kubernetes . . . 492
Solución de problemas y soporte . . . 493
Actualizar Trident . . . 494
Administra backends y volúmenes . . . 495
Solución de problemas . . . 499
Solución de problemas general . . . 499
Despliegue fallido de Trident usando el operador . . . 500
Despliegue fallido de Trident usando `tridentctl` . . . 502
Eliminar por completo Trident y CRDs . . . 502
Fallo de desagrupación de nodo NVMe con espacios de nombres de bloque sin procesar RWX o Kubernetes 1.26 . . . 503
Los clientes de NFSv4.2 informan "argumento no válido" después de actualizar ONTAP cuando esperaban que "v4.2-xattrs" estuviera activado . . . 504
Soporte . . . 504
Ciclo de vida de soporte de Trident . . . 504
Autoservicio . . . 505
Soporte de la comunidad . . . 505
NetApp soporte técnico de NetApp . . . 505
Para más información . . . 505
Referencia . . . 506
Puertos de Trident . . . 506
Descripción general . . . 506
API de REST de Trident . . . 508

Cuándo usar la API de REST . . . 508
Uso de la API de REST . . . 508
Opciones de línea de comandos . . . 509
Registro . . . 509
Kubernetes . . . 509
Docker . . . 510
REST . . . 510
Objetos de Kubernetes y Trident . . . 510
¿Cómo interactúan los objetos entre sí? . . . 510
Kubernetes `PersistentVolumeClaim` objetos . . . 511
Kubernetes `PersistentVolume` objetos . . . 513
Kubernetes `StorageClass` objetos . . . 513
Objetos de Kubernetes `VolumeSnapshotClass` . . . 517
Objetos de Kubernetes `VolumeSnapshot` . . . 517
Objetos de Kubernetes `VolumeSnapshotContent` . . . 518
Objetos de Kubernetes `VolumeGroupSnapshotClass` . . . 518
Objetos de Kubernetes `VolumeGroupSnapshot` . . . 519
Kubernetes `VolumeGroupSnapshotContent` objetos . . . 519
Objetos de Kubernetes `CustomResourceDefinition` . . . 520
Trident `StorageClass` objetos . . . 520
Objetos backend de Trident . . . 520
Objetos de Trident `StoragePool` . . . 521
Objetos de Trident `Volume` . . . 521
Objetos de Trident `Snapshot` . . . 522
Trident `ResourceQuota` objeto . . . 523
Normas de seguridad de los pods (PSS) y Security Context Constraints (SCC) . . . 524
Contexto de seguridad requerido de Kubernetes y campos relacionados . . . 525
Estándares de seguridad de pods (PSS) . . . 525
Políticas de seguridad de pods (PSP) . . . 526
Restricciones de contexto de seguridad (SCC) . . . 527
Avisos legales . . . 530
Copyright . . . 530
Marcas registradas . . . 530
Patentes . . . 530
Política de privacidad . . . 530
Código abierto . . . 530

Documentación de Trident 25.10

Notas de publicación

Novedades

Las notas de la versión proporcionan información sobre nuevas características, mejoras y correcciones de errores en la última versión de NetApp Trident.

> El binario para Linux `tridentctl` que se incluye en el archivo zip del instalador es la versión probada y compatible. Ten en cuenta que el binario `macos` incluido en la `/extras` parte del archivo zip no está probado ni es compatible.

Novedades en 25.10

Conoce las novedades en Trident y Trident Protect, incluyendo mejoras, correcciones y obsolescencias.

Trident

Mejoras

- **Kubernetes**:
  - Se agregó compatibilidad con instantáneas de grupo de volúmenes CSI con las API de Kubernetes de instantáneas de grupo de volúmenes v1beta1 para los controladores ONTAP-NAS NFS y ONTAP-SAN-Economy, además de ONTAP-SAN (iSCSI y FC). Consulta "Trabaja con instantáneas de grupo de volúmenes".
  - Se agregó compatibilidad con la conmutación por error automatizada de cargas de trabajo con desconexión forzada de volumen para ONTAP-NAS y ONTAP-NAS-Economy (excluyendo SMB en ambos controladores NAS), y para los controladores ONTAP-SAN y ONTAP-SAN-Economy. Consulta "Automatizando la conmutación por error de aplicaciones con estado con Trident".
  - Concurrencia de nodos Trident mejorada para mayor escalabilidad en las operaciones de nodos para volúmenes FCP.
  - Se agregó compatibilidad con ONTAP AFX para el controlador ONTAP NAS. Consulta "Opciones de configuración y ejemplos de ONTAP NAS".
  - Se agregó soporte para configurar solicitudes y límites de recursos de CPU y memoria para los contenedores Trident mediante TridentOrchestrator CR y valores del Helm chart. ("Problema #1000", "Problema #927", "Problema #853", "Problema n.º 592", "Problema #110").
  - Se agregó compatibilidad con FC para la personalidad ASAr2. Consulta "Opciones de configuración y ejemplos de ONTAP SAN".
  - Se agregó una opción para mostrar las métricas de Prometheus con HTTPS en lugar de HTTP. Consulta "Monitorea Trident".
  - Se agregó una opción `--no-rename` al importar un volumen para conservar el nombre original pero dejar que Trident administre el ciclo de vida del volumen. Consulta "Importar volúmenes".
  - La implementación de Trident ahora se ejecuta en la clase de prioridad system-cluster-critical.
- Se agregó una opción para que el controlador Trident use la red del host a través de helm, operator y tridentctl ("Problema #858").
- Se agregó soporte manual de QoS al controlador ANF, haciéndolo listo para producción en Trident 25.10; esta mejora experimental se introdujo en Trident 25.06.

Mejoras experimentales

No apto para uso en entornos de producción.

- **[Vista previa técnica]**: se agregó soporte para concurrencia para ONTAP-NAS (solo NFS) y ONTAP-SAN (NVMe para unified ONTAP 9), además de la vista previa técnica existente para el controlador ONTAP-SAN (protocolos iSCSI y FCP en unified ONTAP 9).

Correcciones

- **Kubernetes**:
  - Se solucionó la inconsistencia en el nombre del contenedor node-driver-registrar de CSI estandarizando Linux DaemonSet a node-driver-registrar para que coincida con Windows DaemonSet y la nomenclatura de la imagen del contenedor.
  - Se solucionó un problema donde las políticas de exportación para qtrees heredados no se actualizaban correctamente.
- **Openshift**:
  - Se solucionó que el pod del nodo Trident no se iniciara en los nodos de Windows en Openshift debido a que SCC tenía allowHostDirVolumePlugin configurado en false ("Problema #950").
- Se solucionó que las QPS de la API de Kubernetes no se establecieran a través de Helm ("Problema #975").
- Se solucionó la imposibilidad de montar un Persistent Volume Claim (PVC) basado en una instantánea de un PVC de sistema de archivos XFS basado en NVMe en el mismo nodo de Kubernetes.
- Se solucionó el problema de cambio de UUID después de reiniciar el host/Docker en modo NDVP al agregar nombres de subsistema únicos/compartidos por backend (por ejemplo, netappdvp_subsystem).
- Se corrigieron errores de montaje para volúmenes iSCSI durante la actualización de Trident de versiones anteriores a 23.10 a 24.10 y superiores, resolviendo el problema de "invalid SANType".
- Se solucionó el problema donde el estado del backend de Trident no cambiaba a en línea/fuera de línea sin reiniciar el controlador Trident.
- Se solucionó una condición de carrera intermitente que causaba una ampliación lenta de PVC.
- Se solucionó que las instantáneas no se limpiaban cuando fallaba la clonación de volúmenes.
- Se solucionó el error al deshacer la etapa del volumen cuando su ruta de dispositivo fue cambiada por el kernel.
- Se solucionó el fallo al deshacer la etapa del volumen porque el dispositivo LUKS ya estaba cerrado.
- Se solucionó el problema donde las operaciones de almacenamiento lentas provocaban errores de ContextDeadline.
- El operador de Trident esperará el tiempo de espera configurable de k8s para comprobar la versión de Trident.

Trident Protect

NetApp Trident Protect proporciona capacidades avanzadas de gestión de datos de aplicaciones que mejoran la funcionalidad y la disponibilidad de las aplicaciones de Kubernetes con estado respaldadas por NetApp ONTAP storage systems y el NetApp Trident CSI storage provisioner.

Mejoras

- Se agregaron anotaciones para controlar los tiempos de espera de las CR de instantáneas para las CR de programación y de respaldo:
  - `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`

    Consulta "Anotaciones de backup y programación admitidas".
- Se agregó una anotación al CR de programación para configurar el tiempo de espera de enlace de PVC, que será utilizado por el CR de respaldo: `protect.trident.netapp.io/pvc-bind-timeout-sec`. Consulta "Anotaciones de backup y programación admitidas".
- Mejoras en `tridentctl-protect`las listas de copias de seguridad e instantáneas con un nuevo campo para indicar fallos en el hook de ejecución.

Cambios en 25.06.2

Trident

Correcciones

- **Kubernetes**: Se solucionó un problema crítico donde se detectaban dispositivos iSCSI incorrectos al desconectar volúmenes de los nodos de Kubernetes.

Cambios en 25.06.1

Trident

> Para los clientes que usan SolidFire, por favor no actualices a 25.06.1 debido a un problema conocido al anular la publicación de volúmenes. 25.06.2 se lanzará pronto para solucionar este problema.

Correcciones

- **Kubernetes**:
  - Se solucionó un problema donde los NQN no se verificaban antes de desasignarlos de los subsistemas.
  - Se solucionó un problema donde varios intentos de cerrar un dispositivo LUKS provocaban fallas al separar volúmenes.
  - Se solucionó el desmontaje del volumen iSCSI cuando la ruta del dispositivo cambió desde su creación.
  - Clonación de bloques de volúmenes entre clases de almacenamiento.
- **OpenShift**: Se solucionó un problema por el cual la preparación del nodo iSCSI fallaba con OCP 4.19.
- Se aumentó el tiempo de espera al clonar un volumen usando SolidFire backends ("Problema #1008").

Cambios en 25.06

Trident

Mejoras

- **Kubernetes**:
  - Se agregó compatibilidad para instantáneas de grupos de volúmenes CSI con `v1beta1` Volume Group Snapshot Kubernetes APIs para el controlador ONTAP-SAN iSCSI. Consulta "Trabaja con instantáneas de grupo de volúmenes".

    > VolumeGroupSnapshot es una función beta en Kubernetes con API beta. Kubernetes 1.32 es la versión mínima requerida para VolumeGroupSnapshot.

  - Se agregó compatibilidad con ONTAP ASA r2 para NVMe/TCP además de iSCSI. Consulta "Opciones de configuración y ejemplos de ONTAP SAN".
  - Se agregó compatibilidad con SMB seguro para volúmenes ONTAP-NAS y ONTAP-NAS-Economy. Los usuarios y grupos de Active Directory ahora pueden usarse con volúmenes SMB para mayor seguridad. Consulta "Habilita SMB seguro".
  - Concurrencia de nodos Trident mejorada para mayor escalabilidad en las operaciones de nodos para volúmenes iSCSI.
  - Se agregó `--allow-discards` al abrir volúmenes LUKS para permitir comandos de descarte/TRIM para la reclamación de espacio.
  - Rendimiento mejorado al formatear volúmenes cifrados con LUKS.
  - Limpieza mejorada de LUKS para dispositivos LUKS fallidos pero parcialmente formateados.
  - Idempotencia mejorada del nodo Trident para conectar y desconectar volúmenes NVMe.
  - Se agregó `internalID` un campo a la configuración del volumen Trident para el controlador ONTAP-SAN-Economy.
  - Se agregó compatibilidad con la replicación de volúmenes con SnapMirror para backends NVMe. Consulta "Replica volúmenes usando SnapMirror".

Mejoras experimentales

> No apto para uso en entornos de producción.

- [Vista previa técnica] Se habilitaron operaciones simultáneas del controlador Trident mediante la `--enable-concurrency` feature flag. Esto permite que las operaciones del controlador se ejecuten en paralelo, mejorando el rendimiento en entornos ocupados o grandes.

  > Esta función es experimental y actualmente admite flujos de trabajo paralelos limitados con el controlador ONTAP-SAN (protocolos iSCSI y FCP).

- [Tech Preview] Se agregó soporte manual de QOS con el controlador ANF.

Correcciones

- **Kubernetes**:
  - Se solucionó un problema con CSI NodeExpandVolume donde los dispositivos multivía podían quedar

con tamaños incongruentes cuando los discos SCSI subyacentes no estaban disponibles.
    - Se solucionó el fallo al limpiar políticas de exportación duplicadas para los controladores ONTAP-NAS y ONTAP-NAS-Economy.
    - Se corrigió que los volúmenes GCNV usaran NFSv3 de forma predeterminada cuando `nfsMountOptions` no se configuraba; ahora se admiten los protocolos NFSv3 y NFSv4. Si `nfsMountOptions` no se proporciona, se usará la versión de NFS predeterminada del host (NFSv3 o NFSv4).
    - Se solucionó el problema de implementación al instalar Trident usando Kustomize ("Problema #831").
    - Se corrigieron las políticas de exportación faltantes para las PVC creadas a partir de instantáneas ("Problema #1016").
    - Se solucionó el problema donde los tamaños de volumen ANF no se alinean automáticamente a incrementos de 1 GiB.
    - Problema corregido al usar NFSv3 con Bottlerocket.
- Se solucionó el problema con los volúmenes ONTAP-NAS-Economy que se expandían hasta 300 TB a pesar de fallos al cambiar el tamaño.
- Se solucionó el problema en el que las operaciones de división de clones se realizaban de manera sincrónica al usar la API de REST de ONTAP.

Obsolescencias:

- **Kubernetes**: Se actualizó la versión mínima compatible de Kubernetes a v1.27.

Trident Protect

NetApp Trident Protect proporciona capacidades avanzadas de gestión de datos de aplicaciones que mejoran la funcionalidad y la disponibilidad de las aplicaciones de Kubernetes con estado respaldadas por NetApp ONTAP storage systems y el NetApp Trident CSI storage provisioner.

Mejoras

- Tiempos de restauración mejorados, ofreciendo la opción de hacer copias de seguridad completas más frecuentes.
- Granularidad mejorada de la definición de aplicaciones y restauración selectiva con filtrado de Group-Version-Kind (GVK).
- Resincronización eficiente y replicación inversa al utilizar AppMirrorRelationship (AMR) con NetApp SnapMirror, para evitar la replicación completa de PVC.
- Se agregó la capacidad de usar EKS Pod Identity para crear AppVault buckets, eliminando la necesidad de especificar un secreto con las credenciales del bucket para los clústeres EKS.
- Se agregó la capacidad de omitir la restauración de etiquetas y anotaciones en el espacio de nombres de restauración si es necesario.
- AppMirrorRelationship (AMR) ahora verificará la expansión de PVC de origen y realizará la expansión adecuada en el PVC de destino según sea necesario.

Correcciones

- Se corrigió un error donde los valores de anotación de instantáneas anteriores se aplicaban a instantáneas más nuevas. Ahora todas las anotaciones de instantáneas se aplican correctamente.
- Se define un secreto para el cifrado del data mover (Kopia / Restic) por defecto, si no se define.

- Se agregaron validaciones y mensajes de error mejorados para la creación de S3 appvault.
- AppMirrorRelationship (AMR) ahora solo replica PVs en el estado Bound, para evitar intentos fallidos.
- Se solucionó el problema donde se mostraban errores al obtener AppVaultContent en un AppVault con un gran número de copias de seguridad.
- KubeVirt VMSnapshots se excluyen de las operaciones de restauración y conmutación por error para evitar fallas.
- Se solucionó el problema con Kopia donde las instantáneas se eliminaban prematuramente debido a que la programación de retención predeterminada de Kopia anulaba lo que había establecido el usuario en la programación.

Cambios en 25.02.1

Trident

Correcciones

- **Kubernetes**:
  - Se solucionó un problema en el trident-operator donde los nombres y versiones de imágenes sidecar se completaban incorrectamente cuando se usaba un registro de imágenes no predeterminado ("Problema #983").
  - Se solucionó el problema por el cual las sesiones de multivía no se recuperaban durante una devolución de conmutación por error de ONTAP ("Problema #961").

Cambios en 25.02

A partir de Trident 25.02, el resumen de novedades proporciona detalles sobre mejoras, correcciones y desuso tanto para las versiones de Trident como de Trident Protect.

Trident

Mejoras

- **Kubernetes**:
  - Se añadió soporte para ONTAP ASA r2 para iSCSI.
  - Se agregó compatibilidad para la desconexión forzada de volúmenes ONTAP-NAS durante escenarios de apagado no ordenado de nodos. Los nuevos volúmenes ONTAP-NAS ahora utilizarán políticas de exportación por volumen gestionadas por Trident. Se proporcionó una ruta de actualización para que los volúmenes existentes se adapten al nuevo modelo de política de exportación al cancelar la publicación sin afectar las cargas de trabajo activas.
  - Añadida la anotación cloneFromSnapshot.
  - Se agregó soporte para la clonación de volúmenes entre espacios de nombres.
  - Remediaciones mejoradas de escaneo de reparación automática iSCSI para iniciar nuevos escaneos por host, canal, destino e ID de LUN exactos.
  - Se agregó soporte para Kubernetes 1.32.
- **OpenShift**:
  - Se agregó soporte para la preparación automática de nodos iSCSI para RHCOS en clústeres ROSA.
  - Se agregó soporte para OpenShift Virtualization para controladores ONTAP.

- Se agregó compatibilidad con Fibre Channel en el controlador ONTAP-SAN.
- Se agregó compatibilidad con NVMe LUKS.
- Se cambió a la imagen scratch para todas las imágenes base.
- Se agregó la detección y el registro del estado de conexión iSCSI cuando las sesiones iSCSI deberían iniciar sesión pero no lo hacen ("Problema #961").
- Se agregó soporte para volúmenes SMB con el driver google-cloud-netapp-volumes.
- Se agregó soporte para permitir que los volúmenes ONTAP omitan la cola de recuperación al eliminarse.
- Se agregó soporte para anular imágenes predeterminadas usando SHAs en lugar de etiquetas.
- Se agregó el flag image-pull-secrets al instalador tridentctl.

Correcciones

- **Kubernetes**:
  - Se corrigieron las direcciones IP de nodo faltantes en las políticas de exportación automática ("Problema #965").
  - Se corrigió el cambio prematuro de las políticas de exportación automática a políticas por volumen para ONTAP-NAS-Economy.
  - Corregidas las credenciales de configuración del backend para soportar todas las particiones de AWS ARN disponibles ("Problema #913").
  - Se añadió la opción de desactivar la reconciliación automática del configurador en el operador de Trident ("Problema #924").
  - Se añadió securityContext para el contenedor csi-resizer ("Problema #976").

Trident Protect

NetApp Trident Protect proporciona capacidades avanzadas de gestión de datos de aplicaciones que mejoran la funcionalidad y la disponibilidad de las aplicaciones de Kubernetes con estado respaldadas por NetApp ONTAP storage systems y el NetApp Trident CSI storage provisioner.

Mejoras

- Se añadió soporte de backup y restauración para máquinas virtuales de KubeVirt / OpenShift Virtualization tanto para volumeMode: File como para volumeMode: Block (almacenamiento de dispositivo sin procesar). Este soporte es compatible con todos los drivers de Trident y mejora las características de protección existentes cuando se replica el almacenamiento usando NetApp SnapMirror con Trident Protect.
- Se añadió la capacidad de controlar el comportamiento de congelación a nivel de aplicación para entornos Kubevirt.
- Se añadió compatibilidad para configurar conexiones proxy de AutoSupport.
- Se añadió la capacidad de definir un secreto para el cifrado del data mover (Kopia / Restic).
- Se añadió la capacidad de ejecutar manualmente un gancho de ejecución.
- Se añadió la capacidad de configurar restricciones de contexto de seguridad (SCCs) durante la instalación de Trident Protect.
- Se añadió soporte para configurar nodeSelector durante la instalación de Trident Protect.
- Se añadió compatibilidad con proxy de salida HTTP/HTTPS para objetos AppVault.
- Se amplió ResourceFilter para permitir la exclusión de recursos con alcance de clúster.

- Se añadió compatibilidad con el token de sesión de AWS en las credenciales de S3 AppVault.
- Se añadió compatibilidad para la recopilación de recursos después de los hooks de ejecución previos a la snapshot.

Correcciones

- Mejoramos la gestión de volúmenes temporales para saltar la cola de recuperación de volúmenes de ONTAP.
- Las anotaciones de SCC ahora se restauran a sus valores originales.
- Mejor eficiencia de restauración con soporte para operaciones paralelas.
- Compatibilidad mejorada con los tiempos de espera del gancho de ejecución para aplicaciones más grandes.

Cambios en 24.10.1

Mejoras

- **Kubernetes**: Se añadió compatibilidad con Kubernetes 1.32.
- Se agregó la detección y el registro del estado de conexión iSCSI cuando las sesiones iSCSI deberían iniciar sesión pero no lo hacen ("Problema #961").

Correcciones

- Se corrigieron las direcciones IP de nodo faltantes en las políticas de exportación automática ("Problema #965").
- Se corrigió el cambio prematuro de las políticas de exportación automática a políticas por volumen para ONTAP-NAS-Economy.
- Se actualizaron las dependencias de Trident y Trident-ASUP para abordar CVE-2024-45337 y CVE-2024-45310.
- Se eliminaron los cierres de sesión para portales no CHAP intermitentemente no saludables durante la reparación automática de iSCSI ("Problema #961").

Cambios en 24.10

Mejoras

- El controlador Google Cloud NetApp Volumes ya está disponible de forma general para volúmenes NFS y admite el aprovisionamiento con reconocimiento de zonas.
- GCP Workload Identity se usará como Cloud Identity para Google Cloud NetApp Volumes con GKE.
- Se añadió `formatOptions` el parámetro de configuración a los drivers ONTAP-SAN y ONTAP-SAN-Economy para permitir que los usuarios especifiquen opciones de formato de LUN.
- Se redujo el tamaño mínimo de volumen de Azure NetApp Files a 50 GiB. Se espera que el nuevo tamaño mínimo de Azure esté disponible de forma general en noviembre.
- Se añadió `denyNewVolumePools` un parámetro de configuración para restringir los drivers ONTAP-NAS-Economy y ONTAP-SAN-Economy a pools FlexVol preexistentes.
- Se añadió la detección para la adición, eliminación o cambio de nombre de agregados desde el SVM en todos los controladores ONTAP.

- Se añadieron 18 MiB de sobrecarga a los LUNs LUKS para asegurar que el tamaño de PVC reportado sea utilizable.
- Se mejoró la gestión de errores al preparar y deshacer la preparación de nodos ONTAP-SAN y ONTAP-SAN-Economy para permitir que deshacer la preparación elimine dispositivos después de una preparación fallida.
- Se añadió un generador de roles personalizado que permite a los clientes crear un rol minimalista para Trident en ONTAP.
- Se añadió registro adicional para la resolución de problemas `lsscsi` ("Problema #792").

Kubernetes

- Se añadieron nuevas funciones de Trident para flujos de trabajo nativos de Kubernetes:
  - Protección de datos
  - Migración de datos
  - Recuperación ante desastres
  - Movilidad de aplicaciones

  "Conoce más sobre Trident Protect".
- Se añadió una nueva bandera `--k8s-api-qps` a los instaladores para establecer el valor de QPS que usa Trident para comunicarse con el servidor API de Kubernetes.
- Añadida `--node-prep` la opción a los instaladores para la gestión automática de las dependencias del protocolo de almacenamiento en los nodos del clúster de Kubernetes. Probada y verificada la compatibilidad con el protocolo de almacenamiento iSCSI de Amazon Linux 2023
- Se añadió compatibilidad con la desconexión forzada para volúmenes ONTAP-NAS-Economy durante escenarios de apagado no controlado del nodo.
- Los nuevos volúmenes NFS de ONTAP-NAS-Economy utilizarán políticas de exportación por qtree cuando usen la opción de backend `autoExportPolicy`. Los qtrees solo se asignarán a políticas de exportación restrictivas de nodo en el momento de la publicación para mejorar el control de acceso y la seguridad. Los qtrees existentes se cambiarán al nuevo modelo de política de exportación cuando Trident despublique el volumen de todos los nodos para hacerlo sin afectar las cargas de trabajo activas.
- Se añadió compatibilidad con Kubernetes 1.31.

Mejoras experimentales

- Se añadió una vista previa técnica para el soporte de Fibre Channel en el controlador ONTAP-SAN.

Correcciones

- **Kubernetes**:
  - Corregido el webhook de admisión de Rancher que impedía las instalaciones de Trident Helm ("Problema #839").
  - Clave de Affinity corregida en los valores del helm chart ("Problema #898").
  - Corregido tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector no funcionará con el valor "true" ("Problema #899").
  - Se eliminaron instantáneas efímeras creadas durante la clonación ("Problema #901").
- Se añadió soporte para Windows Server 2019.

- Corregido `go mod tidy` en el repo de Trident ("Problema #767").

Obsolescencias

- **Kubernetes:**
  - Se actualizó la versión mínima compatible de Kubernetes a 1.25.
  - Se eliminó la compatibilidad con POD Security Policy.

Rebranding de producto

A partir de la versión 24.10, Astra Trident pasa a llamarse Trident (NetApp Trident). Este cambio de marca no afecta ninguna función, plataforma compatible ni la interoperabilidad de Trident.

Cambios en 24.06

Mejoras

- **IMPORTANTE**: El parámetro `limitVolumeSize` ahora limita los tamaños de qtree/LUN en los controladores economy de ONTAP. Usa el nuevo parámetro `limitVolumePoolSize` para controlar los tamaños de FlexVol en esos controladores. ("Problema #341").
- Se añadió la capacidad para que la reparación automática de iSCSI inicie escaneos SCSI por ID de LUN exacto si se están usando igroups obsoletos ("Problema #883").
- Se añadió compatibilidad para permitir las operaciones de clonación y cambio de tamaño de volúmenes incluso cuando el backend está en modo suspendido.
- Se añadió la capacidad de que la configuración de registros definida por el usuario para el controlador de Trident se propague a los pods de nodo de Trident.
- Se añadió compatibilidad en Trident para usar REST por defecto en lugar de ONTAPI (ZAPI) para versiones de ONTAP 9.15.1 y posteriores.
- Se añadió compatibilidad para nombres de volúmenes personalizados y metadatos en los backends de almacenamiento ONTAP para nuevos volúmenes persistentes.
- Se mejoró el controlador `azure-netapp-files` (ANF) para habilitar automáticamente el directorio de instantáneas por defecto cuando las opciones de montaje de NFS están configuradas para usar la versión de NFS 4.x.
- Se añadió soporte para Bottlerocket en volúmenes NFS.
- Se añadió compatibilidad en vista previa técnica para Google Cloud NetApp Volumes.

Kubernetes

- Se añadió compatibilidad con Kubernetes 1.30.
- Se añadió la capacidad para que Trident DaemonSet limpie montajes zombie y archivos residuales de seguimiento al iniciar ("Problema #883").
- Se añadió la anotación de PVC `trident.netapp.io/luksEncryption` para importar volúmenes LUKS de forma dinámica ("Problema #849").
- Se añadió reconocimiento de topología al controlador de ANF.
- Se añadió compatibilidad para nodos de Windows Server 2022.

Correcciones

- Solucionados los fallos de instalación de Trident debido a transacciones obsoletas.
- Se corrigió tridentctl para ignorar los mensajes de advertencia de Kubernetes ("Problema #892").
- Cambié la prioridad del controlador Trident `SecurityContextConstraint` a `0` ("Problema #887").
- Los controladores ONTAP ahora aceptan tamaños de volumen inferiores a 20 MiB ("Problema[#885").
- Se corrigió Trident para evitar la contracción de los volúmenes FlexVol durante la operación de redimensionamiento para el controlador ONTAP-SAN.
- Corregido el fallo de importación de volúmenes ANF con NFS v4.1.

Cambios en 24.02

Mejoras

- Se añadió compatibilidad con Cloud Identity.
  - AKS con ANF - Azure Workload Identity se usará como identidad cloud.
  - EKS con FSxN - el rol IAM de AWS se usará como identidad en la nube.
- Se añadió compatibilidad para instalar Trident como un complemento en un clúster de EKS desde la consola de EKS.
- Se añadió la opción de configurar y desactivar la reparación automática de iSCSI ("Problema #864").
- Se añadió la personalidad de Amazon FSx a los drivers de ONTAP para permitir la integración con AWS IAM y SecretsManager, y para que Trident pueda eliminar volúmenes FSx con backups ("Problema #453").

Kubernetes

- Se añadió compatibilidad con Kubernetes 1.29.

Correcciones

- Corregidos los mensajes de aviso de ACP cuando ACP no está activado ("Problema #866").
- Se añadió un retraso de 10 segundos antes de realizar una división de clones durante el borrado de snapshots para los drivers de ONTAP, cuando un clon está asociado con el snapshot.

Obsolescencias

- Se eliminó el marco de atestaciones in-toto de los manifiestos de imagen multiplataforma.

Cambios en 23.10

Correcciones

- Expansión de volumen corregida si un nuevo tamaño solicitado es menor que el tamaño total del volumen para los controladores de almacenamiento ontap-nas y ontap-nas-flexgroup ("Problema #834").
- Tamaño de volumen fijo para mostrar solo el tamaño utilizable del volumen durante la importación para los controladores de almacenamiento ontap-nas y ontap-nas-flexgroup ("Problema #722").
- Corregida la conversión de nombres de FlexVol para ONTAP-NAS-Economy.
- Solucionado el problema de inicialización de Trident en un nodo Windows cuando se reinicia el nodo.

Mejoras

Kubernetes

Se añadió compatibilidad con Kubernetes 1.28.

Trident

- Se añadió compatibilidad para usar Azure Managed Identities (AMI) con el controlador de almacenamiento azure-netapp-files.
- Se añadió compatibilidad con NVMe over TCP para el controlador ONTAP-SAN.
- Se añadió la capacidad de pausar el aprovisionamiento de un volumen cuando el backend está en estado suspendido por el usuario ("Problema #558").

Cambios en 23.07.1

Kubernetes: Se corrigió la eliminación de daemonset para admitir actualizaciones sin tiempo de inactividad ("Problema #740").

Cambios en 23.07

Correcciones

Kubernetes

- Se corrigió la actualización de Trident para que ignore los pods antiguos atascados en estado de terminación ("Problema #740").
- Se añadió tolerancia a la definición de "transient-trident-version-pod" ("Problema #795").

Trident

- Se corrigieron las solicitudes de ONTAPI (ZAPI) para asegurarse de que se consulten los números de serie de LUN al obtener los atributos de LUN, para identificar y solucionar dispositivos iSCSI fantasma durante las operaciones de Node Staging.
- Corregido el manejo de errores en el código del controlador de almacenamiento ("Problema #816").
- Corregido el redimensionamiento de cuotas al usar controladores ONTAP con use-rest=true.
- Corregida la creación de clon LUN en ontap-san-economy.
- Revertir el campo de información de publicación de `rawDevicePath` a `devicePath`; se añadió lógica para rellenar y recuperar (en algunos casos) `devicePath` campo.

Mejoras

Kubernetes

- Se añadió compatibilidad para importar snapshots preaprovisionados.
- Despliegue minimizado y permisos de daemonset linux ("Problema #817").

Trident

- Ya no se informa el campo de estado para volúmenes "online" e instantáneas.

- Actualiza el estado del backend si el backend ONTAP está desconectado ("Problemas #801", "#543").
- El número de serie del LUN siempre se recupera y publica durante el flujo de trabajo ControllerVolumePublish.
- Se añadió lógica adicional para verificar el número de serie y el tamaño del dispositivo iSCSI multivía.
- Verificación adicional para los volúmenes iSCSI para asegurar que el dispositivo de multivía correcto se desmonte.

Mejora experimental

Se añadió compatibilidad en versión preliminar para NVMe over TCP para el controlador ONTAP-SAN.

Documentación

Se han realizado muchas mejoras organizativas y de formato.

Obsolescencias

Kubernetes

- Se eliminó la compatibilidad con snapshots v1beta1.
- Se eliminó la compatibilidad con volúmenes y clases de almacenamiento anteriores a CSI.
- Versión mínima compatible de Kubernetes actualizada a 1.22.

Cambios en 23.04

> La desconexión forzada de volúmenes para ONTAP-SAN-* solo es compatible con versiones de Kubernetes que tengan activada la función Non-Graceful Node Shutdown. La desconexión forzada debe habilitarse durante la instalación usando el `--enable-force-detach` parámetro del instalador de Trident.

Correcciones

- Se corrigió Trident Operator para usar localhost IPv6 para la instalación cuando se especifica en spec.
- Se corrigieron los permisos del rol de clúster del operador Trident para que estén sincronizados con los permisos del paquete ("Problema #799").
- Se solucionó el problema al adjuntar un volumen de bloque sin procesar en varios nodos en modo RWX.
- Se solucionó el soporte de clonación de FlexGroup y la importación de volúmenes para volúmenes SMB.
- Se solucionó el problema donde el controlador Trident no podía apagarse inmediatamente ("Problema #811").
- Se agregó una corrección para mostrar todos los nombres de igroup asociados con un LUN específico aprovisionado con controladores ontap-san-*.
- Se agregó una solución para permitir que los procesos externos se ejecuten hasta su finalización.
- Error de compilación corregido para la arquitectura s390 ("Problema #537").
- Se corrigió el nivel de registro incorrecto durante las operaciones de montaje de volumen ("Problema #781").
- Se corrigió un posible error de aserción de tipo ("Problema #802").

Mejoras

- Kubernetes:
  - Se agregó soporte para Kubernetes 1.27.
  - Se añadió soporte para importar volúmenes LUKS.
  - Se agregó soporte para el modo de acceso PVC ReadWriteOncePod.
  - Se agregó soporte para la desconexión forzada para volúmenes ONTAP-SAN-* durante escenarios de apagado de nodo no elegante.
  - Todos los volúmenes ONTAP-SAN-* ahora usarán igroups por nodo. Los LUN solo se asignarán a igroups mientras se publiquen activamente en esos nodos para mejorar nuestra seguridad. Los volúmenes existentes se migrarán oportunamente al nuevo esquema de igroups cuando Trident determine que es seguro hacerlo sin afectar las cargas de trabajo activas ("Problema #758").
  - Mejoramos la seguridad de Trident limpiando los igroups administrados por Trident que no se usan de los backends ONTAP-SAN-*.
- Se agregó soporte para volúmenes SMB con Amazon FSx a los drivers de almacenamiento ontap-nas-economy y ontap-nas-flexgroup.
- Se agregó soporte para recursos compartidos SMB con los controladores de almacenamiento ontap-nas, ontap-nas-economy y ontap-nas-flexgroup.
- Se agregó soporte para nodos arm64 ("Problema #732").
- Mejoramos el procedimiento de apagado de Trident desactivando primero los servidores API ("Problema #811").
- Se agregó soporte de compilación entre plataformas para hosts Windows y arm64 al Makefile; consulta BUILD.md.

Obsolescencias

Kubernetes: Los igroups con alcance de backend ya no se crearán al configurar los controladores ontap-san y ontap-san-economy ("Problema #758").

Cambios en 23.01.1

Correcciones

- Se corrigió Trident Operator para usar localhost IPv6 para la instalación cuando se especifica en spec.
- Se corrigieron los permisos del rol de clúster de Trident Operator para que estén sincronizados con los permisos del bundle "Problema #799".
- Se agregó una solución para permitir que los procesos externos se ejecuten hasta su finalización.
- Se solucionó el problema al adjuntar un volumen de bloque sin procesar en varios nodos en modo RWX.
- Se solucionó el soporte de clonación de FlexGroup y la importación de volúmenes para volúmenes SMB.

Cambios en 23.01

Kubernetes 1.27 ya es compatible con Trident. Por favor, actualiza Trident antes de actualizar Kubernetes.

Correcciones

- Kubernetes: se agregaron opciones para excluir la creación de Pod Security Policy para corregir las instalaciones de Trident a través de Helm ("Problemas #783, #794").

Mejoras

Kubernetes

- Se agregó soporte para Kubernetes 1.26.
- Mejorada la utilización general de los recursos de Trident RBAC ("Problema #757").
- Se agregó automatización para detectar y arreglar sesiones iSCSI rotas o antiguas en los nodos host.
- Se añadió soporte para expandir volúmenes cifrados LUKS.
- Kubernetes: se agregó compatibilidad con rotación de credenciales para volúmenes cifrados LUKS.

Trident

- Se agregó soporte para volúmenes SMB con Amazon FSx for NetApp ONTAP al controlador de almacenamiento ontap-nas.
- Se agregó soporte para permisos NTFS cuando usas volúmenes SMB.
- Se agregó soporte para grupos de almacenamiento para volúmenes de GCP con nivel de servicio CVS.
- Se agregó soporte para el uso opcional de flexgroupAggregateList al crear FlexGroups con el controlador de almacenamiento ontap-nas-flexgroup.
- Rendimiento mejorado para el controlador de almacenamiento ontap-nas-economy al administrar múltiples FlexVol volumes
- Se habilitaron actualizaciones de dataLIF para todos los controladores de almacenamiento NAS de ONTAP.
- Se actualizó la convención de nomenclatura de implementación de Trident y DaemonSet para reflejar el sistema operativo del nodo host.

Obsolescencias

- Kubernetes: se actualizó la versión mínima compatible de Kubernetes a 1.21.
- Ya no se deben especificar los DataLIFs al configurar `ontap-san` o `ontap-san-economy` controladores.

Cambios en 22.10

Debes leer la siguiente información crítica antes de actualizar a Trident 22.10.

Información crítica sobre Trident 22.10

- Kubernetes 1.25 ya es compatible con Trident. Tienes que actualizar Trident a 22.10 antes de actualizar a Kubernetes 1.25.
- Trident ahora aplica estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

  El uso de una configuración sin multivía o el uso de `find_multipaths: yes` o `find_multipaths: smart` en el archivo multipath.conf provocará fallos de montaje. Trident ha recomendado el uso de `find_multipaths: no` desde la versión 21.07.

Correcciones

- Se solucionó un problema específico del backend de ONTAP creado usando `credentials` que fallaba al ponerse en línea durante la actualización 22.07.0 ("Problema n.º 759").
- **Docker:** Se solucionó un problema que provocaba que el complemento de volumen de Docker no se iniciara en algunos entornos ("Problema #548" y "Problema #760").
- Se solucionó el problema de SLM específico de los backends SAN de ONTAP para garantizar que solo se publique un subconjunto de dataLIFs que pertenecen a los nodos de informes.
- Se solucionó un problema de rendimiento donde ocurrían exploraciones innecesarias de LUN iSCSI al conectar un volumen.
- Se eliminaron los reintentos granulares dentro del flujo de trabajo iSCSI de Trident para fallar rápido y reducir los intervalos de reintentos externos.
- Se solucionó el problema donde se devolvía un error al vaciar un dispositivo iSCSI cuando el dispositivo multivía correspondiente ya estaba vaciado.

Mejoras

- Kubernetes:
  - Se agregó compatibilidad con Kubernetes 1.25. Debes actualizar Trident a 22.10 antes de actualizar a Kubernetes 1.25.
  - Se agregó un ServiceAccount, ClusterRole y ClusterRoleBinding separados para la implementación de Trident y DaemonSet para permitir futuras mejoras de permisos.
  - Se añadió compatibilidad para "uso compartido de volúmenes entre espacios de nombres".
- Todos los controladores de almacenamiento Trident `ontap-*` ahora funcionan con la API de REST de ONTAP.
- Se agregó el nuevo operador yaml (`bundle_post_1_25.yaml`) sin un `PodSecurityPolicy` para admitir Kubernetes 1.25.
- Se agregó "compatibilidad con volúmenes cifrados con LUKS" para `ontap-san` y `ontap-san-economy` controladores de almacenamiento.
- Se agregó soporte para nodos de Windows Server 2019.
- Agregado "Compatibilidad con volúmenes SMB en nodos Windows" a través del `azure-netapp-files` controlador de almacenamiento.
- La detección automática de conmutación de sitios de MetroCluster para los drivers de ONTAP ahora está disponible de forma general.

Obsolescencias

- **Kubernetes:** Se actualizó la versión mínima compatible de Kubernetes a 1.20.
- Se eliminó el controlador Astra Data Store (ADS).
- Se eliminó el soporte para `yes` y `smart` opciones para `find_multipaths` al configurar el multivía del nodo de trabajo para iSCSI.

Cambios en 22.07

Correcciones

Kubernetes

- Se solucionó el problema para manejar valores booleanos y numéricos para el selector de nodos al configurar Trident con Helm o el operador Trident. ("Problema de GitHub #700")
- Se solucionó el problema en el manejo de errores de rutas que no son CHAP, así que kubelet volverá a intentarlo si falla. "Problema de GitHub #736")

Mejoras

- Transición de k8s.gcr.io a registry.k8s.io como registro predeterminado para imágenes CSI
- Los volúmenes ONTAP-SAN ahora usarán igroups por nodo y solo asignarán LUN a igroups mientras se publiquen activamente en esos nodos para mejorar nuestra seguridad. Los volúmenes existentes se cambiarán oportunamente al nuevo esquema de igroups cuando Trident determine que es seguro hacerlo sin afectar las cargas de trabajo activas.
- Se incluyó un ResourceQuota con las instalaciones de Trident para asegurar que Trident DaemonSet se programe cuando el consumo de PriorityClass esté limitado por defecto.
- Se agregó compatibilidad con Network Features al controlador de Azure NetApp Files. ("Problema de GitHub #717")
- Se agregó la detección automática de conmutación de sitios en vista previa técnica de MetroCluster a los controladores ONTAP. ("Problema de GitHub #228")

Obsolescencias

- **Kubernetes:** Se actualizó la versión mínima compatible de Kubernetes a 1.19.
- La configuración de backend ya no permite varios tipos de autenticación en una sola configuración.

Eliminaciones

- Se ha eliminado el controlador AWS CVS (obsoleto desde 22.04).
- Kubernetes
  - Se eliminó la capacidad SYS_ADMIN innecesaria de los pods de nodo.
  - Reduce la preparación del nodo a información simple del host y al descubrimiento de servicios activos para hacer una confirmación de mejor esfuerzo de que los servicios NFS/iSCSI están disponibles en los nodos de trabajo.

Documentación

Se ha agregado una nueva "Estándares de seguridad de pods" sección (PSS) que detalla los permisos habilitados por Trident en la instalación.

Cambios en 22.04

NetApp mejora continuamente y amplía sus productos y servicios. Aquí tienes algunas de las funciones más recientes de Trident. Para versiones anteriores, consulta "Versiones anteriores de la documentación".

Si estás actualizando desde cualquier versión anterior de Trident y usas Azure NetApp Files, el `location` parámetro de configuración ahora es un campo singleton obligatorio.

Correcciones

- Análisis mejorado de los nombres de iniciadores iSCSI. ("Problema de GitHub #681")
- Se solucionó el problema donde no se permitían los parámetros de clase de almacenamiento CSI. ("Problema de GitHub #598")
- Se corrigió la declaración de clave duplicada en Trident CRD. ("Problema de GitHub #671")
- Se corrigieron registros de instantáneas de CSI inexactos. ("Problema de GitHub #629"))
- Se solucionó el problema con la cancelación de publicación de volúmenes en nodos eliminados. ("Problema de GitHub #691")
- Se agregó manejo de inconsistencias del sistema de archivos en dispositivos de bloque. ("Problema de GitHub #656")
- Se solucionó el problema al extraer imágenes de auto-support al establecer la `imageRegistry` flag durante la instalación. ("Problema de GitHub #715")
- Se solucionó el problema donde el controlador de Azure NetApp Files fallaba al clonar un volumen con múltiples reglas de exportación.

Mejoras

- Las conexiones entrantes a los puntos finales seguros de Trident ahora requieren un mínimo de TLS 1.3. ("Problema de GitHub #698")
- Trident ahora añade encabezados HSTS a las respuestas de sus endpoints seguros.
- Trident ahora intenta habilitar automáticamente la función de permisos unix de Azure NetApp Files.
- **Kubernetes**: El daemonset de Trident ahora se ejecuta en la clase de prioridad system-node-critical. ("Problema de GitHub #694")

Eliminaciones

Se ha eliminado el controlador E-Series (deshabilitado desde 20.07).

Cambios en 22.01.1

Correcciones

- Se solucionó el problema con la cancelación de publicación de volúmenes en nodos eliminados. ("Problema de GitHub #691")
- Se solucionó el pánico al acceder a campos nulos para el espacio de agregado en las respuestas de la API de ONTAP.

Cambios en 22.01.0

Correcciones

- **Kubernetes:** Aumenta el tiempo de reintento del registro de nodos para clústeres grandes.
- Se solucionó un problema donde el controlador azure-netapp-files podía confundirse por varios recursos con el mismo nombre.
- Los DataLIF de IPv6 de ONTAP SAN ahora funcionan si se especifican con corchetes.
- Se solucionó el problema donde intentar importar un volumen ya importado devolvía EOF y dejaba el PVC

en estado pendiente. ("Problema de GitHub #489")

- Se solucionó el problema cuando el rendimiento de Trident se ralentizaba cuando se creaban más de 32 instantáneas en un SolidFire volumen.
- Se reemplazó SHA-1 por SHA-256 en la creación de certificados SSL.
- Se corrigió el controlador de Azure NetApp Files para permitir nombres de recursos duplicados y limitar las operaciones a una sola ubicación.
- Se corrigió el controlador de Azure NetApp Files para permitir nombres de recursos duplicados y limitar las operaciones a una sola ubicación.

Mejoras

- Mejoras en Kubernetes:
  - Se añadió compatibilidad con Kubernetes 1.23.
  - Agrega opciones de programación para los pods Trident cuando se instalan mediante Trident Operator o Helm. ("Problema de GitHub #651")
- Permitir volúmenes entre regiones en el controlador de GCP. ("Problema de GitHub #633")
- Se agregó compatibilidad para la opción 'unixPermissions' en los volúmenes de Azure NetApp Files. ("Problema de GitHub #666")

Obsolescencias

La interfaz de REST de Trident solo puede escuchar y servir en las direcciones 127.0.0.1 o [::1]

Cambios en 21.10.1

La versión v21.10.0 tiene un problema que puede hacer que el controlador Trident entre en estado CrashLoopBackOff cuando se elimina un nodo y luego se vuelve a agregar al clúster de Kubernetes. Este problema se soluciona en v21.10.1 (GitHub issue 669).

Correcciones

- Se solucionó una posible condición de carrera al importar un volumen en un backend CVS de GCP que provocaba que fallara la importación.
- Se solucionó un problema que puede poner al controlador Trident en un estado CrashLoopBackOff cuando se elimina un nodo y luego se vuelve a agregar al clúster de Kubernetes (GitHub issue 669).
- Se solucionó el problema donde las SVM ya no se descubrían si no se especificaba ningún nombre de SVM (GitHub issue 612).

Cambios en 21.10.0

Correcciones

- Se solucionó el problema donde los clones de volúmenes XFS no se podían montar en el mismo nodo que el volumen de origen (GitHub issue 514).
- Se solucionó el problema donde Trident registraba un error fatal al apagarse (GitHub issue 597).
- Correcciones relacionadas con Kubernetes:
  - Devuelve el espacio usado de un volumen como el mínimo restoreSize al crear instantáneas con

`ontap-nas` y `ontap-nas-flexgroup` drivers (GitHub issue 645).

- Se solucionó el problema donde `Failed to expand filesystem` se registraba un error después de cambiar el tamaño del volumen (GitHub issue 560).
- Se solucionó el problema donde un pod podía quedarse atascado en `Terminating` state (GitHub issue 572).
- Se solucionó el caso en el que un `ontap-san-economy` FlexVol podría estar lleno de LUN de instantáneas (GitHub issue 533).
- Se solucionó el problema del instalador YAML personalizado con una imagen diferente (GitHub issue 613).
- Cálculo fijo del tamaño de la instantánea (GitHub issue 611).
- Se solucionó el problema donde todos los instaladores de Trident podían identificar Kubernetes simple como OpenShift (GitHub issue 639).
- Se corrigió el operador Trident para detener la conciliación si no se puede acceder al servidor API de Kubernetes (GitHub issue 599).

Mejoras

- Se agregó soporte para la opción `unixPermissions` en los volúmenes de rendimiento de GCP-CVS.
- Se agregó soporte para volúmenes CVS optimizados para escala en GCP en el rango de 600 GiB a 1 TiB.
- Mejoras relacionadas con Kubernetes:
  - Se añadió soporte para Kubernetes 1.22.
  - Se habilitó el operador Trident y el gráfico Helm para que funcionen con Kubernetes 1.22 (GitHub issue 628).
  - Se agregó la imagen del operador al comando `tridentctl` images (GitHub issue 570).

Mejoras experimentales

- Se agregó soporte para la replicación de volumen en el `ontap-san` controlador.
- Se agregó soporte **tech preview** de REST para los `ontap-nas-flexgroup`, `ontap-san` y `ontap-nas-economy` controladores.

Problemas conocidos

Los problemas conocidos identifican problemas que podrían impedirte usar el producto correctamente.

- Al actualizar un clúster de Kubernetes de 1.24 a 1.25 o posterior que tenga Trident instalado, debes actualizar values.yaml para establecer `excludePodSecurityPolicy` en `true` o agregar `--set excludePodSecurityPolicy=true` al comando `helm upgrade` antes de poder actualizar el clúster.
- Trident ahora aplica un espacio en blanco `fsType` (`fsType=""` para los volúmenes que no tienen el `fsType` especificado en su StorageClass. Cuando trabajas con Kubernetes 1.17 o posterior, Trident permite proporcionar un espacio en blanco `fsType` para volúmenes NFS. Para volúmenes iSCSI, tienes que configurar el `fsType` en tu StorageClass cuando aplicas un `fsGroup` usando un Security Context.
- Cuando usas un backend en varias instancias de Trident, cada archivo de configuración de backend debe tener un valor de `storagePrefix` diferente para los backends de ONTAP o usar un `TenantName` diferente para los backends de SolidFire. Trident no puede detectar volúmenes que hayan sido creados por otras instancias de Trident. Si intentas crear un volumen existente en los backends de ONTAP o

SolidFire, funcionará porque Trident trata la creación de volúmenes como una operación idempotente. Si `storagePrefix` o `TenantName` no son diferentes, podría haber conflictos de nombres para los volúmenes creados en el mismo backend.

- Al instalar Trident (usando `tridentctl` o el operador Trident) y usando `tridentctl` para administrar Trident, debes asegurarte de que la variable de entorno `KUBECONFIG` esté configurada. Esto es necesario para indicar el clúster de Kubernetes contra el que `tridentctl` debe trabajar. Cuando trabajes con varios entornos de Kubernetes, debes asegurarte de que el archivo `KUBECONFIG` esté correctamente referenciado.
- Para realizar la reclamación de espacio en línea para PVs iSCSI, el sistema operativo subyacente en el nodo de trabajo podría requerir que se pasen opciones de montaje al volumen. Esto es cierto para las instancias de RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), que requieren el `discard` "opción de montaje"; asegúrate de que la opción discard mountOption esté incluida en tu [`StorageClass`^] para admitir el descarte de bloques en línea.
- Si tienes más de una instancia de Trident por clúster de Kubernetes, Trident no puede comunicarse con otras instancias ni descubrir otros volúmenes que hayan creado, lo que lleva a un comportamiento inesperado e incorrecto si se ejecuta más de una instancia dentro de un clúster. Debe haber solo una instancia de Trident por clúster de Kubernetes.
- Si se eliminan objetos basados en Trident `StorageClass` de Kubernetes mientras Trident está desconectado, Trident no elimina las clases de almacenamiento correspondientes de su base de datos cuando vuelve a estar en línea. Deberías eliminar estas clases de almacenamiento usando `tridentctl` o la API de REST.
- Si un usuario elimina un PV proporcionado por Trident antes de eliminar el PVC correspondiente, Trident no elimina automáticamente el volumen de respaldo. Deberías eliminar el volumen mediante `tridentctl` o la API de REST.
- ONTAP no puede aprovisionar más de un FlexGroup a la vez a menos que el conjunto de agregados sea único para cada solicitud de aprovisionamiento.
- Al usar Trident sobre IPv6, debes especificar `managementLIF` y `dataLIF` en la definición del backend entre corchetes. Por ejemplo, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> No puedes especificar `dataLIF` en un backend SAN de ONTAP. Trident detecta todos los LIF iSCSI disponibles y los usa para establecer la sesión multivía.

- Si usas el `solidfire-san` driver con OpenShift 4.5, asegúrate de que los nodos worker subyacentes usen MD5 como algoritmo de autenticación CHAP. Los algoritmos CHAP seguros compatibles con FIPS SHA1, SHA-256 y SHA3-256 están disponibles con Element 12.7.

Encuentra más información

- "Trident GitHub"
- "Blogs de Trident"

Versiones anteriores de la documentación

Si no estás ejecutando Trident 25.10, la documentación de las versiones anteriores está disponible según el "Ciclo de vida de soporte de Trident".

- "Trident 25.06"

- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"

NetApp soporte Trident para sistemas de almacenamiento ONTAP ASA r2

NetApp Trident 25.02 y versiones posteriores son compatibles con los sistemas NetApp ASA r2 como backends de almacenamiento. Consulta "Sistemas ASA r2" para más información.

Los sistemas ASA r2 requieren el `ontap-san` driver. Trident no admite el `ontap-san-economy` driver para los sistemas ASA r2.

Cuando especificas `ontap-san` como `storageDriverName` en la configuración del backend, Trident detecta automáticamente el sistema de almacenamiento ASA r2.

Trident proporciona protección de datos limitada para los sistemas ASA r2 con Trident protect.

Los protocolos SAN compatibles dependen de tu versión de Trident:

- Trident 25.02 y versiones posteriores son compatibles con iSCSI.
- Trident 25.06 y versiones posteriores admiten NVMe/TCP además de iSCSI.

Debes asignar al menos un agregado a la máquina virtual de almacenamiento (SVM) para el almacenamiento backend de ONTAP. Consulta "Asignar agregados a SVM en sistemas ASA r2" para instrucciones.

Operaciones compatibles

- Aprovisionamiento de volúmenes persistentes (PVs)
- Aprovisionamiento dinámico de volúmenes
- Creación y eliminación de volúmenes
- Clonación de volúmenes
- Expandir volúmenes
- Gestión de clases de almacenamiento

Operaciones no admitidas

- Cifrado LUKS
- SnapMirror replicación de volumen

- Limitar el uso de agregados
- Modo de reserva de espacio
- Instantáneas
- Organización en niveles

Para más información, consulta "Opciones de configuración y ejemplos de ONTAP SAN".

Problemas conocidos

Los problemas conocidos identifican problemas que podrían impedirte usar correctamente esta versión del producto.

Los siguientes problemas conocidos afectan la versión actual:

Restaurar backups de Restic de archivos grandes puede fallar

Al restaurar archivos de 30GB o más desde una copia de seguridad de Amazon S3 hecha con Restic, la operación de restauración puede fallar. Como solución alternativa, haz una copia de seguridad de los datos usando Kopia como data mover (Kopia es el data mover predeterminado para las copias de seguridad). Consulta "Protege las aplicaciones usando Trident Protect" para instrucciones.

Empezar

Conoce más sobre Trident

Conoce más sobre Trident

Trident es un proyecto de código abierto totalmente compatible, mantenido por NetApp. Ha sido diseñado para ayudarte a satisfacer las demandas de persistencia de tu aplicación contenedorizada usando interfaces estándar de la industria, como Container Storage Interface (CSI).

¿Qué es Trident?

Netapp Trident permite el consumo y la gestión de recursos de almacenamiento en todas las plataformas de almacenamiento populares de NetApp, en la nube pública o en las instalaciones, incluidos los clústeres ONTAP locales (AFF, FAS y ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files y Amazon FSx for NetApp ONTAP.

Trident es un orquestador de almacenamiento dinámico compatible con la Interfaz de Almacenamiento de Contenedores (CSI) que se integra de forma nativa con "Kubernetes". Trident se ejecuta como un único Controller Pod más un Node Pod en cada nodo de trabajo del clúster. Consulta "Arquitectura de Trident" para más detalles.

Trident también ofrece integración directa con el ecosistema Docker para las plataformas de almacenamiento de NetApp. El NetApp Docker Volume Plugin (nDVP) admite el aprovisionamiento y la gestión de recursos de almacenamiento desde la plataforma de almacenamiento hasta los hosts Docker. Consulta "Implementa Trident para Docker" para más detalles.

Si es la primera vez que usas Kubernetes, deberías familiarizarte con el "Conceptos y herramientas de Kubernetes".

Integración de Kubernetes con NetApp products

La cartera de productos de almacenamiento de NetApp se integra con muchos aspectos de un clúster de Kubernetes, proporcionando capacidades avanzadas de gestión de datos que mejoran la funcionalidad, la capacidad, el rendimiento y la disponibilidad de la implementación de Kubernetes.

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" es un servicio de AWS completamente administrado que te permite iniciar y ejecutar sistemas de archivos impulsados por el NetApp sistema operativo de almacenamiento ONTAP.

Azure NetApp Files

"Azure NetApp Files" es un servicio para compartir archivos de Azure de clase empresarial, impulsado por NetApp. Puedes ejecutar tus cargas de trabajo basadas en archivos más exigentes en Azure de forma nativa, con el alto rendimiento y la gestión de datos avanzada que esperas de NetApp.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" es un dispositivo de almacenamiento exclusivamente de software que ejecuta el software de gestión ONTAP en la nube.

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes" es un servicio de almacenamiento de archivos totalmente administrado en Google Cloud que proporciona almacenamiento de archivos de alto rendimiento y de clase empresarial.

Software Element

"Elemento" permite al administrador de almacenamiento consolidar cargas de trabajo garantizando el rendimiento y permitiendo un espacio de almacenamiento simplificado y optimizado.

NetApp HCI

"NetApp HCI" simplifica la gestión y la escala del centro de datos al automatizar las tareas rutinarias y permitir que los administradores de infraestructura se concentren en funciones más importantes.

Trident puede aprovisionar y administrar dispositivos de almacenamiento para aplicaciones en contenedores directamente contra la plataforma de almacenamiento NetApp HCI subyacente.

NetApp ONTAP

"NetApp ONTAP" es el NetApp sistema operativo de almacenamiento unificado y multiprotocolo que proporciona capacidades avanzadas de gestión de datos para cualquier aplicación.

Los sistemas ONTAP tienen configuraciones all-flash, híbridas o all-HDD y ofrecen muchos modelos de implementación diferentes: FAS, AFA y ASA clústeres on-premises, ONTAP Select y Cloud Volumes ONTAP. Trident admite estos modelos de implementación de ONTAP.

Arquitectura de Trident

Trident se ejecuta como un único Controller Pod más un Node Pod en cada nodo trabajador del clúster. El node pod debe estar ejecutándose en cualquier host donde quieras montar potencialmente un volumen Trident.

Entender los pods de controlador y los pods de nodo

Trident se despliega como un solo Pod de controlador Trident y uno o más Pods de nodo Trident en el clúster de Kubernetes y usa los *CSI Sidecar Containers* estándar de Kubernetes para simplificar el despliegue de plugins CSI. "Contenedores sidecar de Kubernetes CSI" son mantenidos por la comunidad de Kubernetes Storage.

Kubernetes "selectores de nodos" y "tolerancias y taints" se utilizan para restringir un pod a ejecutarse en un nodo específico o preferido. Puedes configurar selectores de nodo y tolerancias para pods de controlador y

nodo durante la instalación de Trident.

- El complemento del controlador se encarga del aprovisionamiento y la gestión de volúmenes, como las instantáneas y el cambio de tamaño.
- El plugin de nodo se encarga de conectar el almacenamiento al nodo.

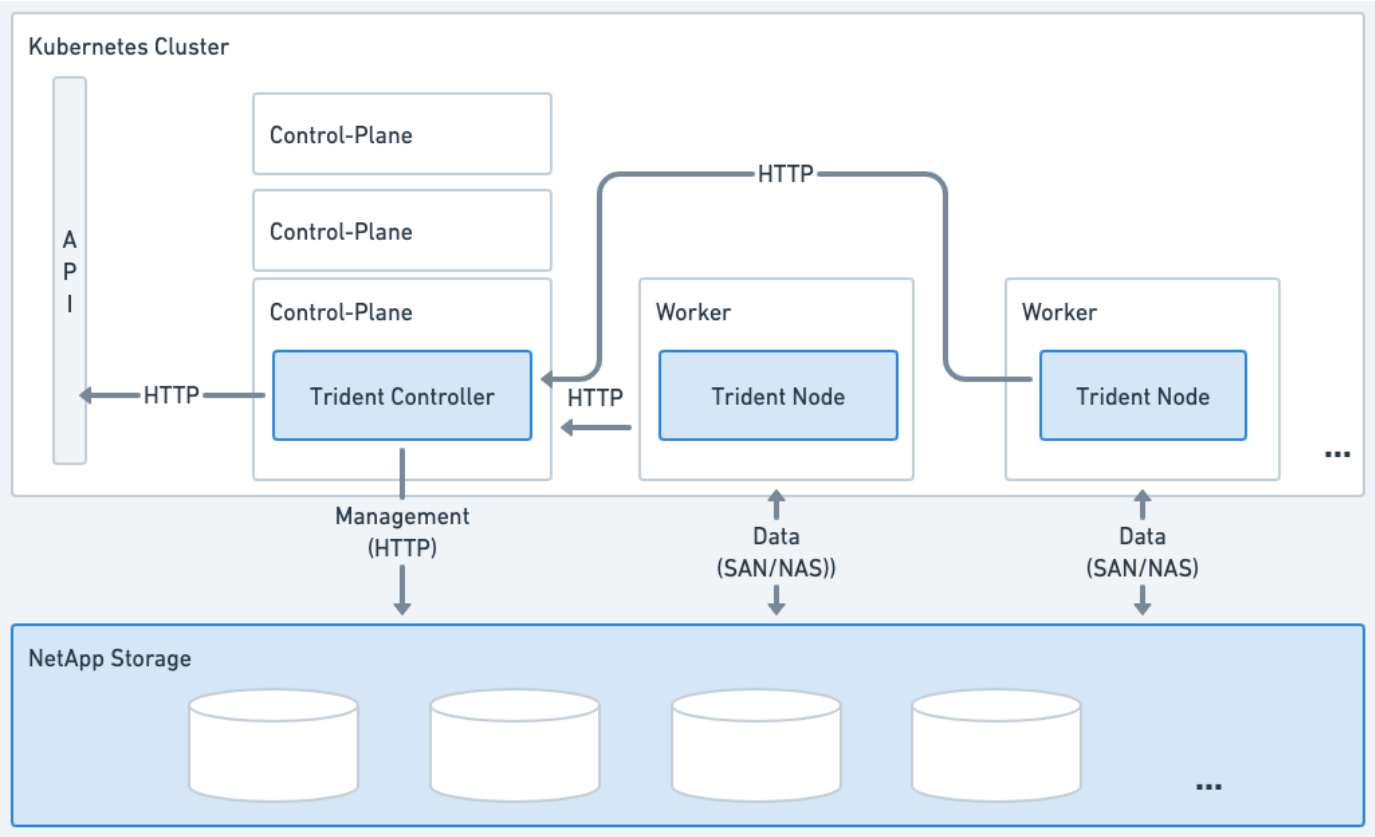


Figura 1. Trident desplegado en el clúster de Kubernetes

Pod de controlador Trident

El pod de controlador Trident es un único pod que ejecuta el complemento de controlador CSI.

- Responsable de aprovisionar y gestionar volúmenes en almacenamiento NetApp
- Gestionado por un despliegue de Kubernetes
- Puede ejecutarse en el plano de control o en los nodos worker, dependiendo de los parámetros de instalación.

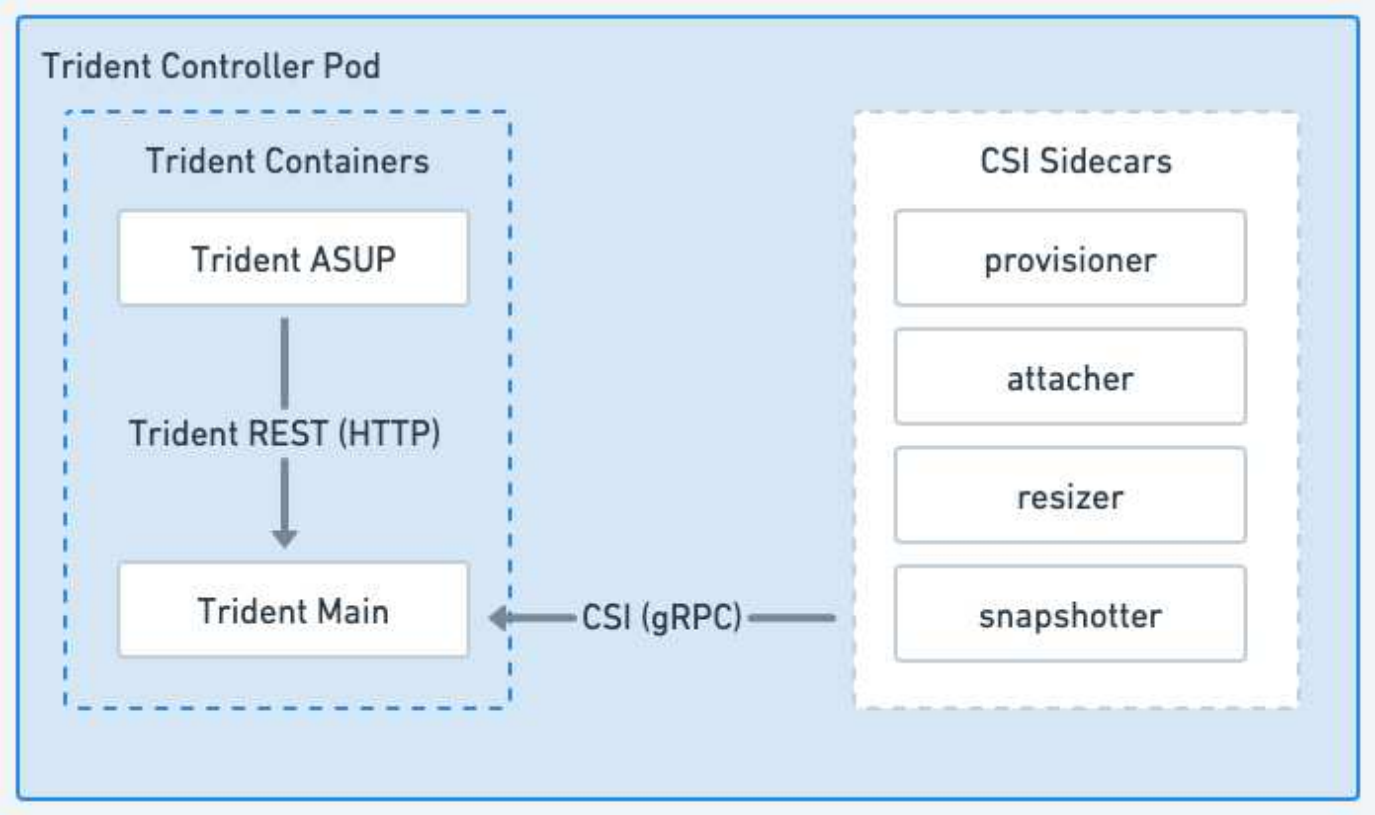


Figura 2. Diagrama del pod del controlador Trident

Pods de nodo Trident

Los pods de nodos Trident son pods privilegiados que ejecutan el plugin de nodos CSI.

- Responsable de montar y desmontar almacenamiento para los Pods que se ejecutan en el host
- Gestionado por un DaemonSet de Kubernetes
- Debe ejecutarse en cualquier nodo que monte almacenamiento NetApp

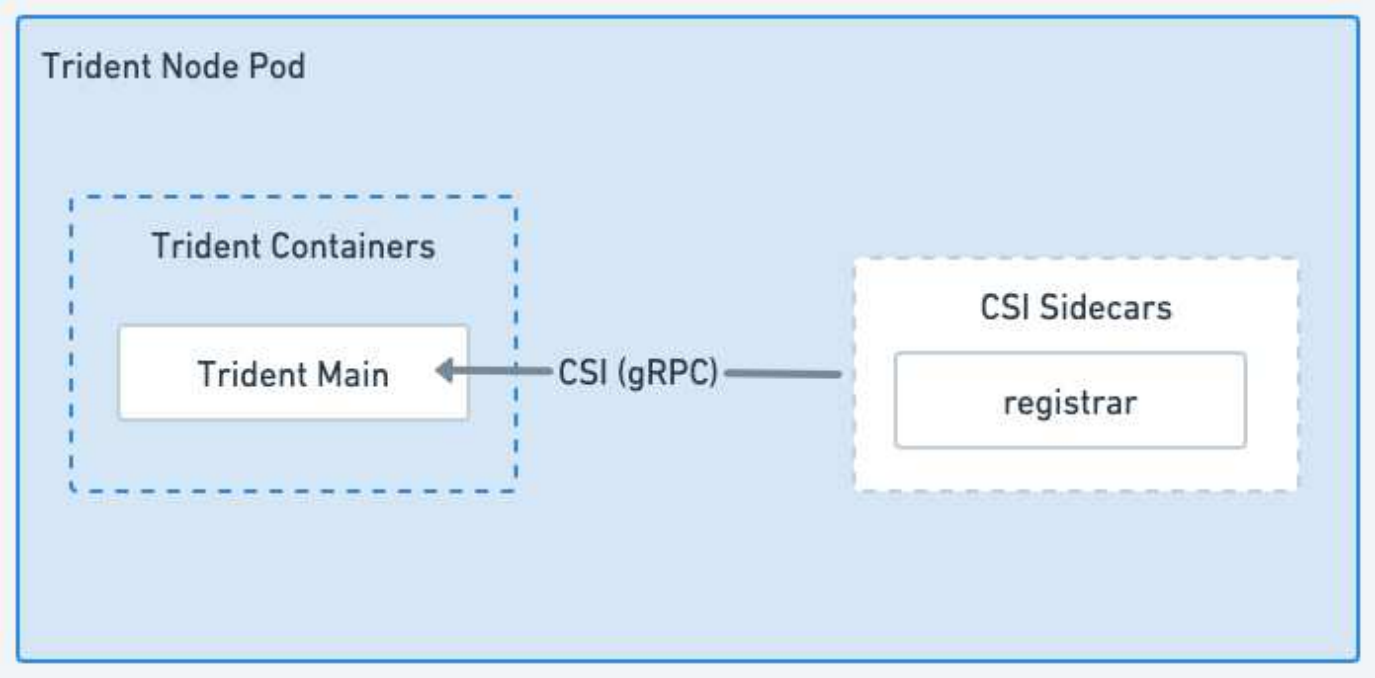


Figura 3. Diagrama del pod del nodo Trident

Arquitecturas de clúster de Kubernetes compatibles

Trident es compatible con las siguientes arquitecturas de Kubernetes:

Arquitecturas de clúster de Kubernetes	Compatible	Instalación predeterminada
Maestro único, compute	Sí	Sí
Múltiples maestros, compute	Sí	Sí
Maestro, `etcd`, computación	Sí	Sí
Maestro, infraestructura, compute	Sí	Sí

Conceptos

Aprovisionamiento

El aprovisionamiento en Trident tiene dos fases principales. La primera fase asocia una clase de almacenamiento con el conjunto de pools de almacenamiento backend adecuados y ocurre como preparación necesaria antes del aprovisionamiento. La segunda fase incluye la creación del volumen en sí y requiere elegir un pool de almacenamiento de entre los asociados a la clase de almacenamiento del volumen pendiente.

Asociación de storage class

La asociación de grupos de almacenamiento backend con una clase de almacenamiento se basa tanto en los atributos solicitados de la clase de almacenamiento como en sus listas `storagePools`,

`additionalStoragePools` y `excludeStoragePools`. Cuando creas una clase de almacenamiento, Trident compara los atributos y los grupos ofrecidos por cada uno de sus backends con los que solicita la clase de almacenamiento. Si los atributos y el nombre de un grupo de almacenamiento coinciden con todos los atributos y nombres de grupo solicitados, Trident añade ese grupo de almacenamiento al conjunto de grupos de almacenamiento adecuados para esa clase de almacenamiento. Además, Trident agrega todos los grupos de almacenamiento que aparecen en la lista `additionalStoragePools` a ese conjunto, incluso si sus atributos no cumplen todos o ninguno de los atributos solicitados por la clase de almacenamiento. Deberías usar la lista `excludeStoragePools` para anular y eliminar grupos de almacenamiento del uso para una clase de almacenamiento. Trident realiza un proceso similar cada vez que agregas un nuevo backend, comprobando si sus grupos de almacenamiento cumplen con los de las clases de almacenamiento existentes y eliminando cualquiera que se haya marcado como excluido.

Creación de volumen

Luego, Trident usa las asociaciones entre las clases de almacenamiento y los pools de almacenamiento para determinar dónde aprovisionar los volúmenes. Cuando creas un volumen, Trident primero obtiene el conjunto de pools de almacenamiento para la clase de almacenamiento de ese volumen y, si especificas un protocolo para el volumen, Trident elimina aquellos pools de almacenamiento que no pueden proporcionar el protocolo solicitado (por ejemplo, un backend NetApp HCI/SolidFire no puede proporcionar un volumen basado en archivos, mientras que un backend ONTAP NAS no puede proporcionar un volumen basado en bloques). Trident aleatoriza el orden de este conjunto resultante para facilitar una distribución uniforme de los volúmenes y luego lo recorre, intentando aprovisionar el volumen en cada pool de almacenamiento por turno. Si lo logra en uno, devuelve el éxito y registra cualquier fallo encontrado en el proceso. Trident devuelve un fallo **solo si** no logra aprovisionar en **todos** los pools de almacenamiento disponibles para la clase de almacenamiento y protocolo solicitados.

Instantáneas de volumen

Conoce más sobre cómo Trident maneja la creación de instantáneas de volumen para sus drivers.

Conoce la creación de instantáneas de volumen

- Para los `ontap-nas`, `ontap-san` y `azure-netapp-files` drivers, cada Persistent Volume (PV) se asigna a un volumen FlexVol. Como resultado, las instantáneas de volumen se crean como instantáneas NetApp. La tecnología de instantáneas NetApp ofrece más estabilidad, escalabilidad, recuperabilidad y rendimiento que las tecnologías de instantáneas de la competencia. Estas copias instantáneas son extremadamente eficientes tanto en el tiempo necesario para crearlas como en el espacio de almacenamiento.
- Para el `ontap-nas-flexgroup` driver, cada Persistent Volume (PV) se asigna a un FlexGroup. Como resultado, las instantáneas de volumen se crean como instantáneas de NetApp FlexGroup. La tecnología de instantáneas de NetApp ofrece más estabilidad, escalabilidad, recuperabilidad y rendimiento que las tecnologías de instantáneas de la competencia. Estas copias instantáneas son extremadamente eficientes tanto en el tiempo necesario para crearlas como en el espacio de almacenamiento.
- Para el `ontap-san-economy` driver, los PV se asignan a LUNs creados en volúmenes compartidos FlexVol. Las VolumeSnapshots de PVs se consiguen realizando FlexClones del LUN asociado. La tecnología ONTAP FlexClone hace posible crear copias incluso de los conjuntos de datos más grandes casi al instante. Las copias comparten bloques de datos con sus padres y no consumen almacenamiento excepto lo necesario para los metadatos.
- Para el `solidfire-san` driver, cada PV se asigna a un LUN creado en el software NetApp Element/NetApp HCI cluster. VolumeSnapshots están representados por instantáneas Element del LUN subyacente. Estas instantáneas son copias de un momento específico y solo ocupan una pequeña

cantidad de recursos y espacio del sistema.

- Cuando trabajas con los controladores `ontap-nas` y `ontap-san`, las instantáneas de ONTAP son copias de un momento específico del FlexVol y consumen espacio en el propio FlexVol. Esto puede hacer que la cantidad de espacio con permisos de escritura en el volumen se reduzca con el tiempo a medida que se crean o programan instantáneas. Una forma sencilla de solucionar esto es aumentar el volumen redimensionándolo desde Kubernetes. Otra opción es eliminar las instantáneas que ya no se necesiten. Cuando se elimina una VolumeSnapshot creada desde Kubernetes, Trident eliminará la instantánea de ONTAP asociada. Las instantáneas de ONTAP que no se crearon desde Kubernetes también se pueden eliminar.

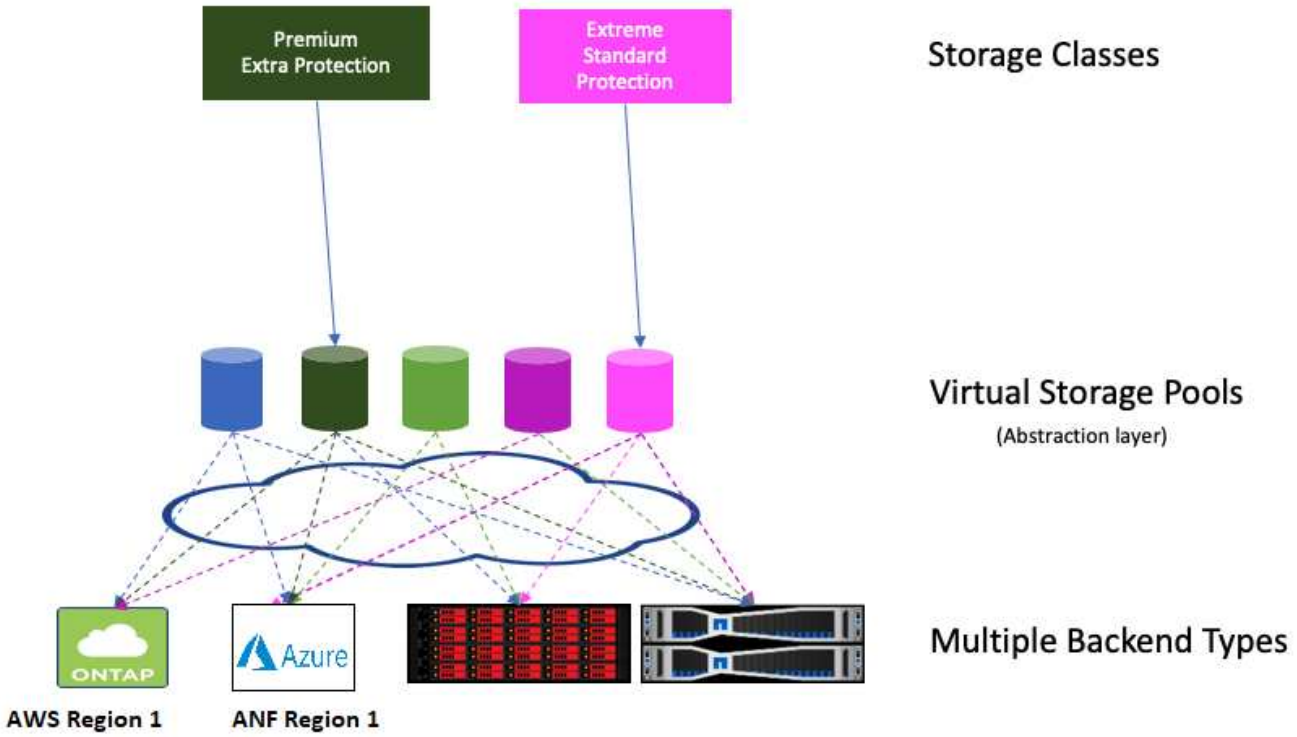
Con Trident, puedes usar VolumeSnapshots para crear nuevos PVs a partir de ellos. Crear PVs a partir de estas instantáneas se realiza usando la tecnología FlexClone para los backends de ONTAP compatibles. Al crear un PV a partir de una instantánea, el volumen de respaldo es un FlexClone del volumen primario de la instantánea. El `solidfire-san` driver usa clones de volumen de software Element para crear PVs a partir de instantáneas. Aquí crea un clon a partir de la instantánea de Element.

Pools virtuales

Los pools virtuales proporcionan una capa de abstracción entre los backends de almacenamiento de Trident y Kubernetes `StorageClasses`. Permiten que un administrador defina aspectos como la ubicación, el rendimiento y la protección para cada backend de una forma común y agnóstica sin hacer que un `StorageClass` especifique qué backend físico, pool de backend o tipo de backend usar para cumplir los criterios deseados.

Conoce los pools virtuales

El administrador de almacenamiento puede definir pools virtuales en cualquiera de los backends de Trident en un archivo de definición JSON o YAML.



Cualquier aspecto especificado fuera de la lista de pools virtuales es global para el backend y se aplicará a todos los pools virtuales, mientras que cada pool virtual puede especificar uno o más aspectos individualmente (anulando cualquier aspecto global del backend).

- Cuando definas pools virtuales, no intentes reorganizar el orden de los pools virtuales existentes en una definición de backend.
- Te recomendamos no modificar los atributos de un pool virtual existente. Deberías definir un nuevo pool virtual para hacer cambios.

La mayoría de los aspectos se especifican en términos específicos del backend. De manera crucial, los valores de los aspectos no se exponen fuera del controlador del backend y no están disponibles para su comparación en `StorageClasses`. En su lugar, el administrador define una o más etiquetas para cada pool virtual. Cada etiqueta es un par clave:valor, y las etiquetas pueden ser comunes a varios backends únicos. Al igual que los aspectos, las etiquetas pueden especificarse por pool o de forma global para el backend. A diferencia de los aspectos, que tienen nombres y valores predefinidos, el administrador tiene total discreción para definir las claves y los valores de las etiquetas según sea necesario. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

Las etiquetas del pool virtual se pueden definir usando estos caracteres:

- letras mayúsculas `A-Z`
- letras minúsculas `a-z`
- números `0-9`
- guiones bajos _

- guiones –

Un `StorageClass` identifica qué pool virtual usar haciendo referencia a las etiquetas dentro de un parámetro selector. Los selectores de pool virtual admiten los siguientes operadores:

Operador	Ejemplo	El valor de la etiqueta de un pool debe:
`=`	performance=premium	Coincidir
`!=`	rendimiento!=extremo	No coincide
`in`	ubicación en (east, west)	Estar en el conjunto de valores
`notin`	performance notin (plata, bronce)	No estar en el conjunto de valores
`<key>`	protección	Existir con cualquier valor
`!<key>`	¡protección!	No existe

Grupos de acceso a volúmenes

Obtén más información sobre cómo Trident usa "grupos de acceso de volúmenes".

> Ignora esta sección si usas CHAP, que se recomienda para simplificar la gestión y evitar el límite de escalado que se describe abajo. Además, si usas Trident en modo CSI, puedes ignorar esta sección. Trident usa CHAP cuando se instala como un aprovisionador CSI mejorado.

Conoce los grupos de acceso de volúmenes

Trident puede utilizar grupos de acceso a volúmenes para controlar el acceso a los volúmenes que aprovisiona. Si CHAP está desactivado, espera encontrar un grupo de acceso llamado `trident` a menos que especifiques uno o más ID de grupo de acceso en la configuración.

Aunque Trident asocia nuevos volúmenes con los grupos de acceso configurados, no crea ni gestiona los grupos de acceso en sí. Los grupos de acceso deben existir antes de que el backend de almacenamiento se agregue a Trident, y necesitan contener los IQN iSCSI de cada nodo en el clúster de Kubernetes que podría montar los volúmenes aprovisionados por ese backend. En la mayoría de las instalaciones, eso incluye cada nodo de trabajo en el clúster.

Para clústeres de Kubernetes con más de 64 nodos, deberías usar varios grupos de acceso. Cada grupo de acceso puede contener hasta 64 IQN, y cada volumen puede pertenecer a cuatro grupos de acceso. Con el máximo de cuatro grupos de acceso configurados, cualquier nodo en un clúster de hasta 256 nodos podrá acceder a cualquier volumen. Para ver los límites más recientes de los grupos de acceso a volúmenes, consulta "aquí".

Si estás modificando la configuración de una que usa el grupo de acceso predeterminado `trident` a una que también usa otros, incluye el ID para el grupo de acceso `trident` en la lista.

Inicio rápido para Trident

Puedes instalar Trident y empezar a gestionar los recursos de almacenamiento en unos pocos pasos. Antes de empezar, revisa "Requisitos de Trident".

Para Docker, consulta "Trident para Docker".

1 Prepara el nodo trabajador

Todos los nodos de trabajo en el clúster de Kubernetes deben poder montar los volúmenes que has aprovisionado para tus pods.

"Prepara el nodo trabajador"

2 Instala Trident

Trident ofrece varios métodos y modos de instalación optimizados para una variedad de entornos y organizaciones.

"Instala Trident"

3 Crear un backend

Un backend define la relación entre Trident y un sistema de almacenamiento. Le dice a Trident cómo comunicarse con ese sistema de almacenamiento y cómo Trident debe aprovisionar volúmenes desde él.

"Configura un backend" para tu sistema de almacenamiento

4 Crea un StorageClass de Kubernetes

El objeto de Kubernetes StorageClass especifica Trident como el aprovisionador y te permite crear una clase de almacenamiento para aprovisionar volúmenes con atributos personalizables. Trident crea una clase de almacenamiento coincidente para los objetos de Kubernetes que especifican el aprovisionador Trident.

"Crear una clase de almacenamiento"

5 Aprovisiona un volumen

Un *PersistentVolume* (PV) es un recurso de almacenamiento físico aprovisionado por el administrador del clúster en un clúster de Kubernetes. El *PersistentVolumeClaim* (PVC) es una solicitud de acceso al PersistentVolume en el clúster.

Crea un PersistentVolume (PV) y un PersistentVolumeClaim (PVC) que usa el StorageClass de Kubernetes configurado para solicitar acceso al PV. Luego puedes montar el PV en un pod.

"Aprovisiona un volumen"

¿Qué sigue?

Ahora puedes agregar backends adicionales, gestionar clases de almacenamiento, gestionar backends y realizar operaciones de volumen.

Requisitos

Antes de instalar Trident, debes revisar estos requisitos generales del sistema. Los backends específicos pueden tener requisitos adicionales.

Información crítica sobre Trident

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.34 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

  El uso de una configuración sin multivía o el uso de `find_multipaths: yes` o `find_multipaths: smart` en el archivo multipath.conf provocará fallos de montaje. Trident ha recomendado el uso de `find_multipaths: no` desde la versión 21.07.

Frontends compatibles (orquestadores)

Trident es compatible con varios motores y orquestadores de contenedores, incluyendo los siguientes:

- Anthos On-Prem (VMware) y Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.34
- OpenShift 4.12, 4.14 - 4.20 (si planeas usar la preparación de nodos iSCSI con OpenShift 4.19, la versión mínima de Trident compatible es 25.06.1.)

  Trident sigue siendo compatible con las versiones anteriores de OpenShift en consonancia con la "Ciclo de vida de las versiones de Red Hat Extended Update Support (EUS)", incluso si dependen de versiones de Kubernetes que ya no tienen soporte oficial upstream. Al instalar Trident en estos casos, puedes ignorar sin problema cualquier mensaje de advertencia sobre la versión de Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.34.x

  Si bien Trident es compatible con Rancher Kubernetes Engine 2 (RKE2) versiones 1.27.x - 1.34.x, Trident actualmente solo ha sido calificado en RKE2 v1.28.5+rke2r1.

Trident también funciona con una serie de otras ofertas de Kubernetes totalmente administradas y autoadministradas, incluyendo Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) y VMWare Tanzu Portfolio.

Trident y ONTAP se pueden usar como proveedor de almacenamiento para "KubeVirt".

Antes de actualizar un clúster de Kubernetes de 1.25 a 1.26 o posterior que tenga Trident instalado, consulta "Actualizar una instalación de Helm".

Backends compatibles (almacenamiento)

Para usar Trident, necesitas uno o más de los siguientes backends compatibles:

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN Array (ASA)
- FAS, AFF o ASA r2 locales (iSCSI, NVMe/TCP y FC) que ejecutan versiones de ONTAP bajo soporte completo o limitado de NetApp. Consulta "Compatibilidad con versiones de software".
- NetApp HCI/Element software 11 o superior

Soporte de Trident para KubeVirt y OpenShift Virtualization

Controladores de almacenamiento compatibles:

Trident admite los siguientes controladores ONTAP para KubeVirt y OpenShift Virtualization:

- ontap-nas
- ontap-nas-economy
- ontap-san (iSCSI, FCP, NVMe sobre TCP)
- ontap-san-economy (solo iSCSI)

Puntos a considerar:

- Actualiza la clase de almacenamiento para que tenga el `fsType` parámetro (por ejemplo: `fsType: "ext4"`) en el entorno de OpenShift Virtualization. Si es necesario, establece el modo de volumen en bloque explícitamente usando el `volumeMode=Block` parámetro en el `dataVolumeTemplates` para notificar a CDI que cree volúmenes de datos de tipo Block.
- *Modo de acceso RWX para controladores de almacenamiento en bloque*: los controladores ontap-san (iSCSI, NVMe/TCP, FC) y ontap-san-economy (iSCSI) solo son compatibles con "volumeMode: Block" (dispositivo sin procesar). Para estos controladores, el `fstype` parámetro no se puede usar porque los volúmenes se proporcionan en modo de dispositivo sin procesar.
- Para flujos de trabajo de migración en vivo donde se requiere el modo de acceso RWX, se admiten estas combinaciones:
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (dispositivo sin procesar)
  - NVMe/TCP + `volumeMode=Block` (dispositivo sin procesar)
  - FC + `volumeMode=Block` (dispositivo sin procesar)

Requisitos de funciones

La siguiente tabla resume las características disponibles con esta versión de Trident y las versiones de Kubernetes que admite.

Característica	Versión de Kubernetes	¿Se requieren feature gates?
Trident	1.27 - 1.34	No
Instantáneas de volumen	1.27 - 1.34	No
PVC a partir de instantáneas de volumen	1.27 - 1.34	No
Redimensionar PV iSCSI	1.27 - 1.34	No
CHAP bidireccional de ONTAP	1.27 - 1.34	No
Políticas dinámicas de exportación	1.27 - 1.34	No
Trident Operator	1.27 - 1.34	No
Topología CSI	1.27 - 1.34	No

Sistemas operativos host probados

Aunque Trident no admite oficialmente sistemas operativos específicos, se sabe que los siguientes funcionan:

- Versiones de Red Hat Enterprise Linux CoreOS (RHCOS) compatibles con OpenShift Container Platform en AMD64 y ARM64
- Red Hat Enterprise Linux (RHEL) 8 o posterior en AMD64 y ARM64

  NVMe/TCP requiere RHEL 9 o posterior.

- Ubuntu 22.04 LTS o posterior en AMD64 y ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 o posterior

De forma predeterminada, Trident se ejecuta en un contenedor y, por lo tanto, se ejecutará en cualquier trabajador de Linux. Sin embargo, esos trabajadores deben poder montar los volúmenes que Trident proporciona usando el cliente de NFS estándar o el iniciador iSCSI, dependiendo de los backends que estés usando.

La `tridentctl` utilidad también funciona en cualquiera de estas distribuciones de Linux.

Configuración del host

Todos los nodos de trabajo del clúster de Kubernetes deben poder montar los volúmenes que has aprovisionado para tus pods. Para preparar los nodos de trabajo, debes instalar herramientas NFS, iSCSI o

NVMe según el controlador que elijas.

"Prepara el nodo trabajador"

Configuración del sistema de almacenamiento

Trident podría requerir cambios en un sistema de almacenamiento antes de que una configuración de backend pueda usarlo.

"Configura backends"

Puertos de Trident

Trident requiere acceso a puertos específicos para comunicarse.

"Puertos de Trident"

Imágenes de contenedores y versiones de Kubernetes correspondientes

Para instalaciones air-gapped, la siguiente lista es una referencia de las imágenes de contenedor necesarias para instalar Trident. Usa el comando `tridentctl images` para verificar la lista de imágenes de contenedor necesarias.

Imágenes de contenedor requeridas para Trident 25.10

Versiones de Kubernetes	Imagen de contenedor
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0	• docker.io/netapp/trident:25.10.0 • docker.io/netapp/trident-autosupport:25.10 • registry.k8s.io/sig-storage/csi-provisioner:v5.3.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v1.14.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.3.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:25.10.0 (opcional)

Instala Trident

Instala usando el operador Trident

Instala usando tridentctl

Instala usando el operador certificado de OpenShift

Usa Trident

Prepara el nodo trabajador

Todos los nodos de trabajo del clúster de Kubernetes deben poder montar los volúmenes que has aprovisionado para tus pods. Para preparar los nodos de trabajo, tienes que instalar herramientas NFS, iSCSI, NVMe/TCP o FC según la selección de tu driver.

Seleccionar las herramientas adecuadas

Si usas una combinación de controladores, deberías instalar todas las herramientas necesarias para tus controladores. Las versiones recientes de Red Hat Enterprise Linux CoreOS (RHCOS) ya traen las herramientas instaladas por defecto.

Herramientas NFS

"Instala las herramientas NFS" si usas: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, o `azure-netapp-files`.

herramientas iSCSI

"Instala las herramientas iSCSI" si estás usando: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Herramientas NVMe

"Instala las herramientas NVMe" si estás usando `ontap-san` para nonvolatile memory express (NVMe) sobre TCP (NVMe/TCP).

> NetApp recomienda ONTAP 9.12 o versiones posteriores para NVMe/TCP.

Herramientas SCSI sobre FC

Consulta "Formas de configurar hosts SAN FC FC-NVMe" para más información sobre cómo configurar tus hosts SAN FC y FC-NVMe.

"Instala las herramientas FC" si estás usando `ontap-san` con sanType `fcp` (SCSI sobre FC).

Puntos a tener en cuenta: * SCSI sobre FC es compatible con los entornos OpenShift y KubeVirt. * SCSI sobre FC no es compatible con Docker. * La reparación automática de iSCSI no es aplicable a SCSI sobre FC.

Herramientas SMB

"Prepárate para aprovisionar volúmenes SMB" si estás usando: `ontap-nas` para aprovisionar volúmenes SMB.

Descubrimiento de servicios de nodo

Trident intenta detectar automáticamente si el nodo puede ejecutar servicios iSCSI o NFS.

> El descubrimiento de servicios de nodo identifica los servicios descubiertos pero no garantiza que los servicios estén configurados correctamente. Por el contrario, la ausencia de un servicio descubierto no garantiza que el montaje del volumen falle.

Revisa los eventos

Trident crea eventos para el nodo para identificar los servicios descubiertos. Para revisar estos eventos, ejecuta:

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

Revisa los servicios descubiertos

Trident identifica los servicios habilitados para cada nodo en el Trident node CR. Para ver los servicios descubiertos, ejecuta:

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

Volúmenes NFS

Instala las herramientas NFS usando los comandos para tu sistema operativo. Asegúrate de que el servicio de NFS se inicie durante el momento de arranque.

RHEL 8+

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

Ubuntu

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

Reinicia tus nodos trabajadores después de instalar las herramientas NFS para evitar fallos al adjuntar volúmenes a los contenedores.

volúmenes iSCSI

Trident puede establecer automáticamente una sesión iSCSI, escanear LUNs, descubrir dispositivos multivía, formatearlos y montarlos en un pod.

capacidades de reparación automática iSCSI

Para los sistemas ONTAP, Trident ejecuta la reparación automática iSCSI cada cinco minutos para:

1. **Identifica** el estado de sesión iSCSI deseado y el estado de sesión iSCSI actual.
2. **Compara** el estado deseado con el estado actual para identificar las reparaciones necesarias. Trident determina las prioridades de reparación y cuándo anticipar las reparaciones.
3. **Realiza las reparaciones** necesarias para devolver el estado de sesión iSCSI actual al estado de sesión iSCSI deseado.

Los registros de la actividad de reparación automática se encuentran en el contenedor `trident-main` en el pod Daemonset correspondiente. Para ver los registros, tienes que haber establecido `debug` en "true" durante la instalación de Trident.

Las capacidades de reparación automática de Trident iSCSI pueden ayudar a prevenir:

- Sesiones iSCSI obsoletas o no saludables que podrían ocurrir después de un problema de conectividad de red. En el caso de una sesión obsoleta, Trident espera siete minutos antes de cerrar la sesión para restablecer la conexión con un portal.

  Por ejemplo, si se rotan los secretos CHAP en el controlador de almacenamiento y la red pierde conectividad, podrían persistir los secretos CHAP antiguos (*stale*). La reparación automática puede reconocer esto y restablecer automáticamente la sesión para aplicar los secretos CHAP actualizados.

- Faltan sesiones iSCSI
- LUNs perdidos

Puntos a tener en cuenta antes de actualizar Trident

- Si solo se utilizan igroups por nodo (introducidos en 23.04+), la reparación automática de iSCSI iniciará rescans SCSI para todos los dispositivos en el bus SCSI.
- Si solo se utilizan igroups con ámbito backend (obsoletos desde la 23.04), la reparación automática de iSCSI iniciará rescans SCSI para los IDs de LUN exactos en el bus SCSI.
- Si se utiliza una mezcla de igroups por nodo e igroups de ámbito backend, la reparación automática de iSCSI iniciará rescans SCSI para LUN IDs exactos en el bus SCSI.

Instala las herramientas iSCSI

Instala las herramientas iSCSI usando los comandos para tu sistema operativo.

Antes de empezar

- Cada nodo en el clúster de Kubernetes debe tener un IQN único. **Este es un requisito previo necesario**.
- Si usas RHCOS versión 4.5 o posterior, u otra distribución de Linux compatible con RHEL, con el `solidfire-san` driver y Element OS 12.5 o anterior, asegúrate de que el algoritmo de autenticación CHAP esté configurado en MD5 en `/etc/iscsi/iscsid.conf`. Los algoritmos CHAP seguros compatibles con FIPS SHA1, SHA-256 y SHA3-256 están disponibles con Element 12.7.

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

- Cuando utilices nodos trabajadores que ejecutan RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con iSCSI PVs, especifica el `discard` mountOption en la StorageClass para realizar la reclamación de espacio en línea. Consulta "Documentación de Red Hat".
- Asegúrate de que has actualizado a la última versión de `multipath-tools`.

RHEL 8+

1. Instala los siguientes paquetes del sistema:

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

2. Verifica que la versión de iscsi-initiator-utils sea 6.2.0.874-2.el7 o posterior:

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

3. Configura el escaneo en manual:

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

4. Activa la multivía:

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

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `iscsid` y `multipathd` están en funcionamiento:

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

6. Activa e inicia `iscsi`:

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

Ubuntu

1. Instala los siguientes paquetes del sistema:

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

2. Verifica que la versión de open-iscsi sea 2.0.874-5ubuntu2.10 o posterior (para bionic) o 2.0.874-7.1ubuntu6.1 o posterior (para focal):

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

3. Configura el escaneo en manual:

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

4. Activa la multivía:

```
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
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `open-iscsi` y `multipath-tools` están habilitados y en funcionamiento:

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

> Para Ubuntu 18.04, debes descubrir los puertos de destino con `iscsiadm` antes de iniciar `open-iscsi` para que el demonio iSCSI se inicie. También puedes modificar el servicio `iscsi` para que se inicie `iscsid` automáticamente.

Configura o desactiva la reparación automática de iSCSI

Puedes configurar las siguientes opciones de reparación automática de Trident iSCSI para arreglar las sesiones obsoletas:

- **intervalo de reparación automática iSCSI**: Determina la frecuencia con la que se invoca la reparación automática iSCSI (por defecto: 5 minutos). Puedes configurarlo para que se ejecute con más frecuencia estableciendo un número menor o con menos frecuencia estableciendo un número mayor.

Establecer el intervalo de reparación automática iSCSI en 0 detiene la reparación automática iSCSI por completo. No recomendamos desactivar la reparación automática iSCSI; solo debería desactivarse en ciertos escenarios cuando la reparación automática iSCSI no funciona como se espera o para fines de depuración.

- **tiempo de espera de reparación automática iSCSI**: determina la duración de la espera de la reparación automática iSCSI antes de cerrar la sesión no saludable e intentar volver a iniciar sesión (por defecto: 7 minutos). Puedes configurarlo con un número mayor para que las sesiones que se identifiquen como no saludables tengan que esperar más tiempo antes de cerrar la sesión y luego se intente volver a iniciar sesión, o con un número menor para cerrar la sesión e iniciarla antes.

Helm

Para configurar o cambiar los ajustes de reparación automática de iSCSI, pasa los parámetros `iscsiSelfHealingInterval` y `iscsiSelfHealingWaitTime` durante la instalación o actualización de helm.

El siguiente ejemplo establece el intervalo de reparación automática iSCSI en 3 minutos y el tiempo de espera de reparación automática en 6 minutos:

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Para configurar o cambiar los ajustes de reparación automática de iSCSI, pasa los parámetros `iscsi-self-healing-interval` y `iscsi-self-healing-wait-time` durante la instalación o actualización de tridentctl.

El siguiente ejemplo establece el intervalo de reparación automática iSCSI en 3 minutos y el tiempo de espera de reparación automática en 6 minutos:

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

Volúmenes NVMe/TCP

Instala las herramientas NVMe usando los comandos para tu sistema operativo.

- NVMe requiere RHEL 9 o posterior.
- Si la versión del kernel de tu nodo de Kubernetes es demasiado antigua o si el paquete NVMe no está disponible para tu versión del kernel, puede que tengas que actualizar la versión del kernel de tu nodo a una que tenga el paquete NVMe.

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
```

Verifica la instalación

Después de la instalación, verifica que cada nodo del clúster de Kubernetes tenga un NQN único usando el comando:

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

Trident modifica el valor de `ctrl_device_tmo` para asegurarse de que NVMe no abandone la ruta si se cae. No cambies esta configuración.

Volúmenes SCSI sobre FC

Ahora puedes usar el protocolo Fibre Channel (FC) con Trident para aprovisionar y gestionar recursos de almacenamiento en el sistema ONTAP.

Prerrequisitos

Configura los ajustes de red y de nodo necesarios para FC.

Configuración de red

1. Obtén el WWPN de las interfaces de destino. Consulta "network interface show" para más información.
2. Obtén el WWPN de las interfaces en el iniciador (host).

   Consulta las utilidades correspondientes del sistema operativo host.

3. Configura la división en zonas en el conmutador FC usando los WWPN del host y del target.

   Consulta la documentación del proveedor del conmutador correspondiente para más información.

   Consulta la siguiente documentación de ONTAP para más detalles:

   - "Visión general de la división en zonas Fibre Channel y FCoE"
   - "Formas de configurar hosts SAN FC FC-NVMe"

Instala las herramientas FC

Instala las herramientas de FC usando los comandos para tu sistema operativo.

- Cuando uses nodos trabajadores que ejecutan RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con FC PVs, especifica el `discard` mountOption en la StorageClass para realizar la reclamación de espacio en línea. Consulta "Documentación de Red Hat".

RHEL 8+

1. Instala los siguientes paquetes del sistema:

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

2. Activa la multivía:

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

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipathd` está funcionando:

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

Ubuntu

1. Instala los siguientes paquetes del sistema:

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

2. Activa la multivía:

```
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
```

> Asegúrate de que `/etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipath-tools` está activado y en funcionamiento:

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

Prepárate para aprovisionar volúmenes SMB

Puedes aprovisionar volúmenes SMB usando `ontap-nas` drivers.

> Debes configurar tanto los protocolos NFS como SMB/CIFS en la SVM para crear un `ontap-nas-economy` volumen SMB para clústeres ONTAP locales. Si no configuras alguno de estos protocolos, la creación del volumen SMB fallará.

> `autoExportPolicy` no es compatible con volúmenes SMB.

Antes de empezar

Antes de que puedas aprovisionar volúmenes SMB, debes tener lo siguiente.

- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident admite volúmenes SMB montados en pods que se ejecutan solo en nodos de Windows.
- Al menos un secreto de Trident con tus credenciales de Active Directory. Para generar un secreto `smbcreds`:

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

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

Pasos

1. Para ONTAP local, puedes crear opcionalmente un recurso compartido SMB o Trident puede crear uno para ti.

   > Se requieren recursos compartidos SMB para Amazon FSx for ONTAP.

   Puedes crear los recursos compartidos de administrador de SMB de dos maneras: usando el complemento "Microsoft Management Console" Shared Folders o usando la CLI de ONTAP. Para crear los recursos compartidos de SMB usando la CLI de ONTAP:

   a. Si es necesario, crea la estructura de ruta del directorio para el recurso compartido.

      El `vserver cifs share create`comando comprueba la ruta especificada en la opción -path durante la creación del recurso compartido. Si la ruta especificada no existe, el comando falla.

   b. Crea un recurso compartido SMB asociado con la SVM especificada:

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

c. Verifica que se creó el recurso compartido:

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

Consulta "Crear un recurso compartido SMB" para obtener detalles completos.

2. Al crear el backend, debes configurar lo siguiente para especificar los volúmenes SMB. Para todas las opciones de configuración del backend de FSx for ONTAP, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

Parámetro	Descripción	Ejemplo
`smbShare`	Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco.	`smb-share`
`nasType`	**Debe establecerse en `smb`.** Si es nulo, el valor predeterminado es `nfs`.	`smb`
`securityStyle`	Estilo de seguridad para nuevos volúmenes. **Debe configurarse en `ntfs` o `mixed` para volúmenes SMB.**	`ntfs` o `mixed` para volúmenes SMB
`unixPermissions`	Modo para nuevos volúmenes. **Debe dejarse vacío para volúmenes SMB.**	""

Configura y gestiona backends

Configura backends

Un backend define la relación entre Trident y un sistema de almacenamiento. Le dice a Trident cómo comunicarse con ese sistema de almacenamiento y cómo Trident debe aprovisionar volúmenes desde él.

Trident ofrece automáticamente grupos de almacenamiento de backends que cumplen los requisitos definidos por una clase de almacenamiento. Aprende cómo configurar el backend para tu sistema de almacenamiento.

- "Configura un backend de Azure NetApp Files"
- "Configura un backend de Google Cloud NetApp Volumes"
- "Configura un backend de NetApp HCI o SolidFire"
- "Configura un backend con los drivers NAS de ONTAP o Cloud Volumes ONTAP"
- "Configura un backend con controladores SAN ONTAP o Cloud Volumes ONTAP"

- "Usa Trident con Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configura un backend de Azure NetApp Files

Puedes configurar Azure NetApp Files como el backend para Trident. Puedes conectar volúmenes NFS y SMB usando un backend de Azure NetApp Files. Trident también admite la gestión de credenciales usando identidades administradas para clústeres de Azure Kubernetes Services (AKS).

Detalles del controlador de Azure NetApp Files

Trident proporciona los siguientes controladores de almacenamiento de Azure NetApp Files para comunicarse con el clúster. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`azure-netapp-files`	NFS SMB	Sistema de archivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Consideraciones

- El servicio Azure NetApp Files no admite volúmenes menores de 50 GiB. Trident crea automáticamente volúmenes de 50 GiB si se solicita un volumen menor.
- Trident admite volúmenes SMB montados en pods que se ejecutan solo en nodos Windows.

Identidades administradas para AKS

Trident es compatible "identidades gestionadas" con clústeres de Azure Kubernetes Services. Para aprovechar la administración optimizada de credenciales que ofrecen las identidades administradas, tienes que tener:

- Un clúster de Kubernetes implementado mediante AKS
- Identidades administradas configuradas en el clúster de kubernetes de AKS
- Trident instalado que incluye el `cloudProvider` para especificar `"Azure"`.

Operador de Trident

Para instalar Trident usando el operador Trident, edita `tridentorchestrator_cr.yaml` para establecer `cloudProvider` en `"Azure"`. Por ejemplo:

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

Helm

El siguiente ejemplo instala conjuntos de Trident `cloudProvider` en Azure usando la variable de entorno `$CP`:

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

El siguiente ejemplo instala Trident y establece la `cloudProvider` bandera en `Azure`:

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

Identidad en la nube para AKS

La identidad en la nube permite que los pods de Kubernetes accedan a los recursos de Azure autenticándose como una identidad de carga de trabajo en lugar de proporcionar credenciales explícitas de Azure.

Para aprovechar la identidad en la nube en Azure, debes tener:

- Un clúster de Kubernetes implementado mediante AKS
- Identidad de carga de trabajo y oidc-issuer configurados en el clúster AKS Kubernetes
- Trident instalado que incluye la `cloudProvider` para especificar `"Azure"` y `cloudIdentity` especificando la identidad de la carga de trabajo

Operador de Trident

Para instalar Trident usando el operador Trident, edita `tridentorchestrator_cr.yaml` para establecer `cloudProvider` en `"Azure"` y establece `cloudIdentity` en `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Por ejemplo:

```
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

Establece los valores para las banderas **cloud-provider (CP)** y **cloud-identity (CI)** usando las siguientes variables de entorno:

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

El siguiente ejemplo instala Trident y configura `cloudProvider` en Azure usando la variable de entorno `$CP` y configura el `cloudIdentity` usando la variable de entorno `$CI`:

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

<code>tridentctl</code>

Establece los valores para los indicadores **cloud provider** y **cloud identity** usando las siguientes variables de entorno:

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

El siguiente ejemplo instala Trident y establece la `cloud-provider` bandera en `$CP`, y `cloud-identity` en `$CI`:

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

Prepárate para configurar un backend de Azure NetApp Files

Antes de que puedas configurar tu backend de Azure NetApp Files, necesitas asegurarte de que se cumplan los siguientes requisitos.

Requisitos previos para volúmenes NFS y SMB

Si usas Azure NetApp Files por primera vez o en una nueva ubicación, necesitas hacer una configuración inicial para preparar Azure NetApp Files y crear un volumen NFS. Consulta "Azure: configura Azure NetApp Files y crea un volumen NFS".

Para configurar y usar un "Azure NetApp Files"backend, necesitas lo siguiente:

> - `subscriptionID`, `tenantID`, `clientID`, `location` y `clientSecret` son opcionales cuando usas identidades administradas en un clúster de AKS.
> - `tenantID`, `clientID` y `clientSecret` son opcionales cuando usas una identidad en la nube en un clúster de AKS.

- Un fondo de capacidad. Consulta "Microsoft: crea un grupo de capacidad para Azure NetApp Files".
- Una subred delegada a Azure NetApp Files. Consulta "Microsoft: delega una subred a Azure NetApp Files".
- `subscriptionID` desde una suscripción de Azure con Azure NetApp Files habilitado.
- `tenantID`, `clientID` y `clientSecret` de un "Registro de la aplicación" en Azure Active Directory con permisos suficientes para el servicio Azure NetApp Files. El registro de la aplicación debe usar una de las siguientes opciones:
  - El rol de Owner o Contributor "predefinido por Azure".
  - A "rol de colaborador personalizado" en el nivel de suscripción (`assignableScopes`) con los siguientes permisos que están limitados solo a lo que Trident requiere. Después de crear el rol personalizado, "asigna el rol usando el portal de Azure".

Rol de colaborador personalizado

```
{
  "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": []
        }
      ]
    }
}
```

- El Azure `location` que contiene al menos un "subred delegada". Desde Trident 22.01, el parámetro `location` es un campo obligatorio en el nivel superior del archivo de configuración del backend. Se ignoran los valores de ubicación especificados en los grupos virtuales.
- Para usar `Cloud Identity`, consigue el `client ID` de un "identidad administrada asignada por el usuario" y especifica ese ID en `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisitos adicionales para volúmenes SMB

Para crear un volumen SMB, debes tener:

- Active Directory configurado y conectado a Azure NetApp Files. Consulta "Microsoft: crear y administrar conexiones de Active Directory para Azure NetApp Files".
- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident admite volúmenes SMB montados en pods que se ejecutan solo en nodos de Windows.
- Al menos un secreto de Trident que contenga tus credenciales de Active Directory para que Azure NetApp Files pueda autenticarse en Active Directory. Para generar el secreto `smbcreds`:

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

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

Opciones y ejemplos de configuración del backend de Azure NetApp Files

Conoce las opciones de configuración de backend de NFS y SMB para Azure NetApp Files y revisa ejemplos de configuración.

Opciones de configuración del backend

Trident usa tu configuración de backend (subred, red virtual, nivel de servicio y ubicación) para crear volúmenes de Azure NetApp Files en grupos de capacidad que estén disponibles en la ubicación solicitada y coincidan con el nivel de servicio y la subred solicitados.

Los backends de Azure NetApp Files proporcionan estas opciones de configuración.

Parámetro	Descripción	Predeterminado
`version`		Siempre 1
`storageDriverName`	Nombre del controlador de almacenamiento	"azure-netapp-files"
`backendName`	Nombre personalizado o el backend de almacenamiento	Nombre del driver + "_" + caracteres aleatorios
`subscriptionID`	El identificador de suscripción de tu suscripción de Azure opcional cuando las identidades administradas están habilitadas en un clúster de AKS.	
`tenantID`	El ID del inquilino de un App Registration es opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS.	
`clientID`	El ID de cliente de un registro de aplicación, opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS.	
`clientSecret`	El secreto del cliente de un registro de aplicación es opcional cuando se usan identidades administradas o identidad en la nube en un clúster de AKS.	
`serviceLevel`	Uno de `Standard`, `Premium` o `Ultra`	"" (aleatorio)
`location`	Nombre de la ubicación de Azure donde se crearán los nuevos volúmenes Opcional cuando las identidades administradas están habilitadas en un clúster de AKS.	
`resourceGroups`	Lista de grupos de recursos para filtrar recursos descubiertos	"[]" (sin filtro)

Parámetro	Descripción	Predeterminado
`netappAccounts`	Lista de cuentas de NetApp para filtrar recursos descubiertos	"[]" (sin filtro)
`capacityPools`	Lista de grupos de capacidad para filtrar recursos descubiertos	"[]" (sin filtro, aleatorio)
`virtualNetwork`	Nombre de una red virtual con una subred delegada	""
`subnet`	Nombre de una subred delegada a `Microsoft.Netapp/volumes`	""
`networkFeatures`	Conjunto de funciones de red virtual para un volumen, puede ser `Basic` o `Standard`. Network Features no está disponible en todas las regiones y podría tener que habilitarse en una suscripción. Especificar `networkFeatures` cuando la funcionalidad no está habilitada hace que el aprovisionamiento del volumen falle.	""
`nfsMountOptions`	Control preciso de las opciones de montaje de NFS. Ignorado para volúmenes SMB. Para montar volúmenes usando la versión de NFS 4.1, incluye `nfsvers=4` en la lista de opciones de montaje separadas por comas para elegir NFS v4.1. Las opciones de montaje establecidas en una definición de clase de almacenamiento reemplazan las opciones de montaje establecidas en la configuración del backend.	"nfsvers=3"
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor	"" (no aplicado por defecto)
`debugTraceFlags`	Indicadores de depuración para usar cuando estés resolviendo problemas. Ejemplo, `\{"api": false, "method": true, "discovery": true}`. No uses esto a menos que estés resolviendo problemas y necesites un volcado detallado del registro.	null

Parámetro	Descripción	Predeterminado
`nasType`	Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo configuras en null, se usan volúmenes NFS por defecto.	`nfs`
`supportedTopologies`	Representa una lista de regiones y zonas compatibles con este backend. Para más información, consulta "Usa la topología CSI".	
`qosType`	Representa el tipo de QoS: automático o manual.	Automático
`maxThroughput`	Establece el rendimiento máximo permitido en MiB/s. Compatible solo con grupos de capacidad de QoS manual.	`4 MiB/sec`

Para obtener más información sobre las funciones de red, consulta "Configura funciones de red para un volumen de Azure NetApp Files".

Permisos y recursos necesarios

Si recibes el error "No capacity pools found" al crear una PVC, es probable que el registro de tu app no tenga los permisos y recursos necesarios (subred, red virtual, capacity pool) asociados. Si la depuración está habilitada, Trident registrará los recursos de Azure detectados cuando se cree el backend. Verifica que se esté usando un rol apropiado.

Los valores para `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` y `subnet` se pueden especificar usando nombres cortos o nombres completamente calificados. Se recomiendan los nombres completamente calificados en la mayoría de las situaciones, ya que los nombres cortos pueden coincidir con varios recursos con el mismo nombre.

Si la vNet está ubicada en un grupo de recursos diferente de la cuenta de almacenamiento de Azure NetApp Files (ANF), especifica el grupo de recursos para la red virtual mientras configuras la lista resourceGroups para el backend.

Los `resourceGroups`, `netappAccounts`, y `capacityPools` valores son filtros que restringen el conjunto de recursos detectados a los disponibles para este backend de almacenamiento y pueden especificarse en cualquier combinación. Los nombres completos siguen este formato:

Tipo	Formato
Grupo de recursos	<resource group>
Cuenta de NetApp	<resource group>/<netapp account>
Fondo de capacidad	<resource group>/<netapp account>/<capacity pool>
Red virtual	<resource group>/<virtual network>
Subred	<resource group>/<virtual network>/<subnet>

Aprovisionamiento de volumen

Puedes controlar el aprovisionamiento de volúmenes predeterminados especificando las siguientes opciones en una sección especial del archivo de configuración. Consulta Configuraciones de ejemplo para más detalles.

Parámetro	Descripción	Predeterminado
`exportRule`	Reglas de exportación para volúmenes nuevos. `exportRule` debe ser una lista separada por comas de cualquier combinación de direcciones IPv4 o subredes IPv4 en notación CIDR. Se ignora para volúmenes SMB.	"0.0.0.0/0"
`snapshotDir`	Controla la visibilidad del directorio .snapshot	"verdadero" para NFSv4 "falso" para NFSv3
`size`	El tamaño predeterminado de los nuevos volúmenes	"100G"
`unixPermissions`	Los permisos unix de los nuevos volúmenes (4 dígitos octales). Se ignora para volúmenes SMB.	"" (función de vista previa, requiere whitelisting en la suscripción)

Configuraciones de ejemplo

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

Configuración mínima

Esta es la configuración mínima absoluta del backend. Con esta configuración, Trident detecta todas tus cuentas NetApp, grupos de capacidad y subredes delegadas a Azure NetApp Files en la ubicación configurada, y coloca los nuevos volúmenes en uno de esos grupos y subredes aleatoriamente. Porque `nasType` se omite, `nfs` se aplica el valor predeterminado y el backend aprovisionará volúmenes NFS.

Esta configuración es ideal cuando recién estás empezando a usar Azure NetApp Files y estás probando cosas, pero en la práctica vas a querer proporcionar un alcance adicional para los volúmenes que aprovisiones.

```
---
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
```

Identidades administradas para AKS

Esta configuración de backend omite `subscriptionID`, `tenantID`, `clientID` y `clientSecret`, que son opcionales cuando usas identidades administradas.

```
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
```

Identidad en la nube para AKS

Esta configuración de backend omite `tenantID`, `clientID` y `clientSecret`, que son opcionales cuando usas una identidad en la nube.

```
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
```

Configuración específica del nivel de servicio con filtros de grupo de capacidad

Esta configuración de backend coloca volúmenes en la `eastus` ubicación de Azure en un `Ultra` grupo de capacidad. Trident detecta automáticamente todas las subredes delegadas a Azure NetApp Files en esa ubicación y coloca un nuevo volumen en una de ellas aleatoriamente.

```
---
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
```

Ejemplo de backend con grupos de capacidad QoS manuales

Esta configuración de back-end coloca volúmenes en la `eastus` ubicación de Azure con grupos de capacidad de QoS manuales.

```
---
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
```

Configuración avanzada

Esta configuración de backend reduce aún más el alcance de la ubicación del volumen a una sola subred, y también modifica algunos valores predeterminados de aprovisionamiento de volumen.

```
---
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"
```

Configuración de virtual pool

Esta configuración de backend define varios grupos de almacenamiento en un solo archivo. Esto es útil cuando tienes varios grupos de capacidad que admiten diferentes niveles de servicio y quieres crear clases de almacenamiento en Kubernetes que los representen. Se usaron etiquetas de grupo virtual para diferenciar los grupos según `performance`.

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

Configuración de topologías admitidas

Trident facilita el aprovisionamiento de volúmenes para cargas de trabajo según regiones y zonas de disponibilidad. El `supportedTopologies` bloque en esta configuración de backend se usa para proporcionar una lista de regiones y zonas por backend. Los valores de región y zona especificados aquí deben coincidir con los valores de región y zona de las etiquetas en cada nodo del clúster de Kubernetes. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en una clase de almacenamiento. Para las clases de almacenamiento que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea volúmenes en la región y zona mencionadas. Para más información, consulta "Usa la topología CSI".

```
---
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
```

Definiciones de clases de almacenamiento

Las siguientes `StorageClass` definiciones se refieren a los grupos de almacenamiento anteriores.

Ejemplos de definiciones usando `parameter.selector` field

Usando `parameter.selector` puedes especificar para cada `StorageClass`el pool virtual que se usa para alojar un volumen. El volumen tendrá los aspectos definidos en el pool elegido.

```
---
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
```

Definiciones de ejemplo para volúmenes SMB

Usando `nasType`, `node-stage-secret-name` y `node-stage-secret-namespace`, puedes especificar un volumen SMB y proporcionar las credenciales de Active Directory necesarias.

Configuración básica en el espacio de nombres predeterminado

```
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"
```

Usando diferentes secretos por espacio de nombres

```
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}
```

Usando diferentes secretos por volumen

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

`nasType: smb`filtros para grupos que admiten volúmenes SMB. `nasType: nfs` o `nasType: null`filtros para grupos NFS.

Crea el backend

Después de crear el archivo de configuración de backend, ejecuta el siguiente comando:

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

Si la creación del backend falla, algo está mal con la configuración del backend. Puedes ver los logs para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs
```

Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando create.

Google Cloud NetApp Volumes

Configura un backend de Google Cloud NetApp Volumes

Ahora puedes configurar Google Cloud NetApp Volumes como backend para Trident. Puedes conectar volúmenes NFS y SMB usando un backend de Google Cloud NetApp Volumes.

Detalles del controlador de Google Cloud NetApp Volumes

Trident proporciona el `google-cloud-netapp-volumes` driver para comunicarse con el clúster. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`google-cloud-netapp-volumes`	NFS SMB	Sistema de archivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Identidad cloud para GKE

La identidad en la nube permite que los pods de Kubernetes accedan a los recursos de Google Cloud autenticándose como una identidad de carga de trabajo en vez de proporcionar credenciales explícitas de Google Cloud.

Para aprovechar la identidad en la nube en Google Cloud, debes tener:

- Un clúster de Kubernetes implementado usando GKE.
- Identidad de carga de trabajo configurada en el clúster de GKE y servidor MetaData de GKE configurado en los grupos de nodos.

- Una cuenta de servicio de GCP con el rol de Google Cloud NetApp Volumes Admin (roles/netapp.admin) o un rol personalizado.
- Trident instalado que incluye el cloudProvider para especificar "GCP" y cloudIdentity especificando la nueva cuenta de servicio de GCP. A continuación se muestra un ejemplo.

Operador de Trident

Para instalar Trident usando el operador Trident, edita `tridentorchestrator_cr.yaml` para establecer `cloudProvider` en `"GCP"` y establece `cloudIdentity` en `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`.

Por ejemplo:

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

Helm

Establece los valores para las banderas **cloud-provider (CP)** y **cloud-identity (CI)** usando las siguientes variables de entorno:

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

El siguiente ejemplo instala Trident y establece `cloudProvider` en GCP usando la variable de entorno `$CP` y establece el `cloudIdentity` usando la variable de entorno `$ANNOTATION`:

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

<code>tridentctl</code>

Establece los valores para los indicadores **cloud provider** y **cloud identity** usando las siguientes variables de entorno:

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

El siguiente ejemplo instala Trident y establece la `cloud-provider` bandera en `$CP`, y `cloud-identity` en `$ANNOTATION`:

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

Prepárate para configurar un backend de Google Cloud NetApp Volumes

Antes de que puedas configurar tu backend de Google Cloud NetApp Volumes, necesitas asegurarte de que se cumplan los siguientes requisitos.

Requisitos previos para volúmenes NFS

Si usas Google Cloud NetApp Volumes por primera vez o en una nueva ubicación, se requiere una configuración inicial para configurar Google Cloud NetApp Volumes y crear un volumen NFS. Consulta "Antes de empezar".

Asegúrate de tener lo siguiente antes de configurar el backend de Google Cloud NetApp Volumes:

- Una cuenta de Google Cloud configurada con el servicio Google Cloud NetApp Volumes. Consulta "Google Cloud NetApp Volumes".
- Número de proyecto de tu cuenta de Google Cloud. Consulta "Identificación de proyectos".
- Una cuenta de servicio de Google Cloud con el rol de NetApp Volumes Admin (`roles/netapp.admin`. Consulta "Roles y permisos de Identity and Access Management".
- Archivo de clave API para tu cuenta GCNV. Consulta "Crear una clave de cuenta de servicio"
- Un grupo de almacenamiento. Consulta "Descripción general de los storage pools".

Para obtener más información sobre cómo configurar el acceso a Google Cloud NetApp Volumes, consulta "Configura el acceso a Google Cloud NetApp Volumes".

Opciones y ejemplos de configuración del backend de Google Cloud NetApp Volumes

Conoce las opciones de configuración de backend para Google Cloud NetApp Volumes y revisa ejemplos de configuración.

Opciones de configuración del backend

Cada backend aprovisiona volúmenes en una única región de Google Cloud. Para crear volúmenes en otras regiones, puedes definir backends adicionales.

Parámetro	Descripción	Predeterminado
`version`		Siempre 1
`storageDriverName`	Nombre del controlador de almacenamiento	El valor de `storageDriverName` debe especificarse como "google-cloud-netapp-volumes".
`backendName`	(Opcional) Nombre personalizado del backend de almacenamiento	Nombre del driver + "_" + parte de la clave API

Parámetro	Descripción	Predeterminado
`storagePools`	Parámetro opcional usado para especificar grupos de almacenamiento para la creación de volúmenes.	
`projectNumber`	Número de proyecto de la cuenta de Google Cloud. El valor se encuentra en la página de inicio del portal de Google Cloud.	
`location`	La ubicación de Google Cloud donde Trident crea volúmenes GCNV. Al crear clústeres de Kubernetes entre regiones, los volúmenes creados en un `location` se pueden usar en cargas de trabajo programadas en nodos de varias regiones de Google Cloud. El tráfico entre regiones tiene un costo adicional.	
`apiKey`	Clave API para la cuenta de servicio de Google Cloud con la `netapp.admin` función. Incluye el contenido en formato JSON del archivo de clave privada de una cuenta de servicio de Google Cloud (copiado textualmente en el archivo de configuración del backend). El `apiKey` debe incluir pares clave-valor para las siguientes claves: `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` y `client_x509_cert_url`.	
`nfsMountOptions`	Control detallado de las opciones de montaje de NFS.	"nfsvers=3"
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor.	"" (no aplicado por defecto)
`serviceLevel`	El nivel de servicio de un pool de almacenamiento y sus volúmenes. Los valores son `flex`, `standard`, `premium` o `extreme`.	
`labels`	Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes	""
`network`	Red de Google Cloud utilizada para volúmenes GCNV.	
`debugTraceFlags`	Indicadores de depuración para usar cuando estés resolviendo problemas. Ejemplo, `{"api":false, "method":true}`. No uses esto a menos que estés resolviendo problemas y necesites un volcado detallado del registro.	null
`nasType`	Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo configuras en null, se usan volúmenes NFS por defecto.	`nfs`

Parámetro	Descripción	Predeterminado
`supportedTopologies`	Representa una lista de regiones y zonas compatibles con este backend. Para más información, consulta "Usa la topología CSI". Por ejemplo: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Opciones de aprovisionamiento de volumen

Puedes controlar el aprovisionamiento de volúmenes predeterminados en la sección `defaults` del archivo de configuración.

Parámetro	Descripción	Predeterminado
`exportRule`	Las reglas de exportación para nuevos volúmenes. Debe ser una lista separada por comas de cualquier combinación de direcciones IPv4.	"0.0.0.0/0"
`snapshotDir`	Acceso al `.snapshot` directorio	"verdadero" para NFSv4 "falso" para NFSv3
`snapshotReserve`	Porcentaje de volumen reservado para instantáneas	"" (acepta el valor predeterminado de 0)
`unixPermissions`	Los permisos unix de nuevos volúmenes (4 dígitos octales).	""

Configuraciones de ejemplo

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

Configuración mínima

Esta es la configuración mínima absoluta del backend. Con esta configuración, Trident detecta todos tus grupos de almacenamiento delegados a Google Cloud NetApp Volumes en la ubicación configurada y coloca los nuevos volúmenes en uno de esos grupos aleatoriamente. Porque `nasType` se omite, la `nfs` configuración predeterminada se aplica y el backend aprovisionará volúmenes NFS.

Esta configuración es ideal cuando recién estás comenzando a utilizar Google Cloud NetApp Volumes y estás probando cosas, pero en la práctica lo más probable es que necesites proporcionar un alcance adicional para los volúmenes que aprovisiones.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----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
```

Configuración para volúmenes SMB

```
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
```

Configuración con filtro StoragePools

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----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
```

Configuración de virtual pool

Esta configuración de backend define varios grupos virtuales en un solo archivo. Los grupos virtuales se definen en la `storage` sección. Son útiles cuando tienes varios grupos de almacenamiento que admiten diferentes niveles de servicio y quieres crear clases de almacenamiento en Kubernetes que los representen. Las etiquetas de los grupos virtuales se usan para diferenciarlos. Por ejemplo, en el ejemplo de abajo, la `performance` etiqueta y el `serviceLevel` tipo se usan para diferenciar los grupos virtuales.

También puedes establecer algunos valores predeterminados que se aplicarán a todos los grupos virtuales y sobrescribir esos valores predeterminados para grupos virtuales individuales. En el siguiente ejemplo, `snapshotReserve` y `exportRule` sirven como valores predeterminados para todos los grupos virtuales.

Para más información, consulta "Pools virtuales".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----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
```

Identidad cloud para 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
```

Configuración de topologías admitidas

Trident facilita el aprovisionamiento de volúmenes para cargas de trabajo según regiones y zonas de disponibilidad. El `supportedTopologies` bloque en esta configuración de backend se usa para proporcionar una lista de regiones y zonas por backend. Los valores de región y zona especificados aquí deben coincidir con los valores de región y zona de las etiquetas en cada nodo del clúster de Kubernetes. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en una clase de almacenamiento. Para las clases de almacenamiento que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea volúmenes en la región y zona mencionadas. Para más información, consulta "Usa la topología CSI".

```
---
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
```

¿Qué sigue?

Después de crear el archivo de configuración de backend, ejecuta el siguiente comando:

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

Para verificar que el backend se haya creado correctamente, ejecuta el siguiente comando:

```
kubectl get tridentbackendconfig

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

Si falla la creación del backend, algo anda mal con la configuración del backend. Puedes describir el backend usando el `kubectl get tridentbackendconfig <backend-name>` comando o ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs
```

Después de identificar y corregir el problema con el archivo de configuración, puedes eliminar el backend y ejecutar el comando create otra vez.

Definiciones de clases de almacenamiento

La siguiente es una `StorageClass`definición básica que se refiere al backend de arriba.

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

Ejemplos de definiciones usando el `parameter.selector` campo:

Usando `parameter.selector` puedes especificar para cada `StorageClass` el "pool virtual" que se utiliza para alojar un volumen. El volumen tendrá los aspectos definidos en el pool elegido.

```
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
```

Para más detalles sobre las clases de almacenamiento, consulta "Crear una clase de almacenamiento".

Definiciones de ejemplo para volúmenes SMB

Usando `nasType`, `node-stage-secret-name` y `node-stage-secret-namespace`, puedes especificar un volumen SMB y proporcionar las credenciales necesarias de Active Directory. Cualquier usuario/contraseña de Active Directory, con o sin permisos, puede usarse para el secreto de la etapa del nodo.

Configuración básica en el espacio de nombres predeterminado

```
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"
```

Usando diferentes secretos por espacio de nombres

```
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}
```

Usando diferentes secretos por volumen

```
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`filtros para grupos que admiten volúmenes SMB. `nasType: nfs` o `nasType: null`filtros para grupos NFS.

Ejemplo de definición de PVC

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

Para verificar si el PVC está enlazado, ejecuta el siguiente comando:

```
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
```

Configura un backend de NetApp HCI o SolidFire

Aprende cómo crear y usar un backend de Element con tu instalación de Trident.

Detalles del controlador Element

Trident proporciona el `solidfire-san` controlador de almacenamiento para comunicarse con el clúster. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

El `solidfire-san` controlador de almacenamiento admite los modos de volumen *file* y *block*. Para el `Filesystem` volumeMode, Trident crea un volumen y crea un sistema de archivos. El tipo de sistema de archivos se especifica mediante el StorageClass.

Controlador	Protocolo	VolumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`solidfire-san`	iSCSI	Bloque	RWO, ROX, RWX, RWOP	Sin sistema de archivos. Dispositivo de bloque sin formato.
`solidfire-san`	iSCSI	Sistema de archivos	RWO, RWOP	`xfs`, `ext3`, `ext4`

Antes de empezar

Necesitarás lo siguiente antes de crear un backend de Element.

- Un sistema de almacenamiento compatible que ejecuta Element software.
- Credenciales para un administrador o usuario inquilino de un clúster NetApp HCI/SolidFire que pueda administrar volúmenes.
- Todos tus nodos de trabajo de Kubernetes deben tener instaladas las herramientas iSCSI adecuadas. Consulta "Información de preparación del nodo worker".

Opciones de configuración del backend

Consulta la siguiente tabla para ver las opciones de configuración del backend:

Parámetro	Descripción	Predeterminado
`version`		Siempre 1
`storageDriverName`	Nombre del controlador de almacenamiento	Siempre "solidfire-san"
`backendName`	Nombre personalizado o el backend de almacenamiento	"solidfire_" + dirección IP de almacenamiento (iSCSI)
`Endpoint`	MVIP para el clúster SolidFire con credenciales de inquilino	
`SVIP`	Dirección IP y puerto de almacenamiento (iSCSI)	
`labels`	Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes.	""
`TenantName`	Nombre del tenant a usar (se crea si no se encuentra)	
`InitiatorIFace`	Restringe el tráfico iSCSI a una interfaz del host específica	"default"
`UseCHAP`	Usa CHAP para autenticar iSCSI. Trident usa CHAP.	verdadero

Parámetro	Descripción	Predeterminado
`AccessGroups`	Lista de ID de grupos de acceso para usar	Encuentra el ID de un grupo de acceso llamado "trident"
`Types`	Especificaciones de QoS	
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor	"" (no aplicado por defecto)
`debugTraceFlags`	Indicadores de depuración para usar al solucionar problemas. Por ejemplo, {"api":false, "method":true}	null

No uses `debugTraceFlags` a menos que estés solucionando problemas y necesites un volcado de registro detallado.

Ejemplo 1: configuración de backend para `solidfire-san` driver con tres tipos de volumen

Este ejemplo muestra un archivo backend que utiliza autenticación CHAP y modela tres tipos de volúmenes con garantías de QoS específicas. Lo más probable es que luego definas clases de almacenamiento para consumir cada uno de estos usando el `IOPS` parámetro de clase de almacenamiento.

```
---
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
```

Ejemplo 2: configuración de backend y clase de almacenamiento para `solidfire-san`driver con grupos virtuales

Este ejemplo muestra el archivo de definición de backend configurado con grupos virtuales junto con StorageClasses que hacen referencia a ellos.

Trident copia las etiquetas presentes en un pool de almacenamiento al LUN de almacenamiento backend durante el aprovisionamiento. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En el archivo de definición de backend de ejemplo que se muestra a continuación, se establecen valores predeterminados específicos para todos los grupos de almacenamiento, que establecen el `type` en Silver. Los grupos virtuales se definen en la sección `storage`. En este ejemplo, algunos grupos de almacenamiento establecen su propio tipo y otros anulan los valores predeterminados establecidos arriba.

```
---
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
```

Las siguientes definiciones de StorageClass se refieren a los grupos virtuales mencionados arriba. Usando el campo `parameters.selector`, cada StorageClass indica qué grupo(s) virtual(es) se pueden usar para

alojar un volumen. El volumen tendrá los aspectos definidos en el grupo virtual elegido.

El primer StorageClass (`solidfire-gold-four`) se asignará al primer pool virtual. Este es el único pool que ofrece rendimiento gold con un `Volume Type QoS` de Gold. El último StorageClass (`solidfire-silver`) identifica cualquier pool de almacenamiento que ofrezca un rendimiento silver. Trident decidirá qué pool virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

```
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
```

Encuentra más información

- "Grupos de acceso a volúmenes"

Controladores SAN de ONTAP

Descripción general del controlador SAN de ONTAP

Conoce cómo configurar un backend de ONTAP con los controladores SAN de ONTAP y Cloud Volumes ONTAP.

Detalles del controlador SAN de ONTAP

Trident proporciona los siguientes controladores de almacenamiento SAN para comunicarse con el clúster ONTAP. Los modos de acceso admitidos son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`ontap-san`	iSCSI SCSI sobre FC	Bloque	RWO, ROX, RWX, RWOP	Sin sistema de archivos; dispositivo de bloque sin formato
`ontap-san`	iSCSI SCSI sobre FC	Sistema de archivos	RWO, RWOP ROX y RWX no están disponibles en el modo de volumen de sistema de archivos.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consulta Consideraciones adicionales para NVMe/TCP.	Bloque	RWO, ROX, RWX, RWOP	Sin sistema de archivos; dispositivo de bloque sin formato

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`ontap-san`	NVMe/TCP Consulta Consideraciones adicionales para NVMe/TCP.	Sistema de archivos	RWO, RWOP ROX y RWX no están disponibles en el modo de volumen de sistema de archivos.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Bloque	RWO, ROX, RWX, RWOP	Sin sistema de archivos; dispositivo de bloque sin formato
`ontap-san-economy`	iSCSI	Sistema de archivos	RWO, RWOP ROX y RWX no están disponibles en el modo de volumen de sistema de archivos.	`xfs`, `ext3`, `ext4`

- Usa `ontap-san-economy` solo si esperas que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP".
- Usa `ontap-nas-economy` solo si se espera que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP" y no se puede usar el controlador `ontap-san-economy`.
- No uses `ontap-nas-economy` si crees que vas a necesitar protección de datos, recuperación ante desastres o movilidad.
- NetApp no recomienda usar FlexVol autogrow en todos los controladores ONTAP, excepto ontap-san. Como solución alternativa, Trident admite el uso de snapshot reserve y escala los volúmenes FlexVol en consecuencia.

Permisos de usuario

Trident espera ejecutarse como administrador de ONTAP o SVM, normalmente usando el `admin`usuario del clúster o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. Para las implementaciones de Amazon FSx for NetApp ONTAP, Trident espera ejecutarse como administrador de ONTAP o SVM, usando el usuario del clúster `fsxadmin` o un `vsadmin`usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. El `fsxadmin`usuario es un reemplazo limitado para el usuario administrador del clúster.

Si usas el parámetro `limitAggregateUsage`, se requieren permisos de administrador de clúster. Cuando usas Amazon FSx for NetApp ONTAP con Trident, el parámetro `limitAggregateUsage` no funcionará con las cuentas de usuario `vsadmin` y `fsxadmin`. La operación de configuración fallará si especificas este parámetro.

Aunque es posible crear un rol más restrictivo dentro de ONTAP que un controlador Trident pueda usar, no lo recomendamos. La mayoría de las nuevas versiones de Trident llamarán a APIs adicionales que habría que

tener en cuenta, lo que hace que las actualizaciones sean difíciles y propensas a errores.

Consideraciones adicionales para NVMe/TCP

Trident admite el protocolo non-volatile memory express (NVMe) usando el `ontap-san` driver, incluyendo:

- IPv6
- Instantáneas y clones de volúmenes NVMe
- Cambiar el tamaño de un volumen NVMe
- Importar un volumen NVMe que se creó fuera de Trident para que su ciclo de vida pueda ser gestionado por Trident
- Multipathing nativo de NVMe
- Apagado correcto o incorrecto de los nodos K8s (24.06)

Trident no admite:

- DH-HMAC-CHAP que es compatible de forma nativa con NVMe
- Mapeador de dispositivos (DM) multipathing
- Cifrado LUKS

> NVMe solo es compatible con las API REST de ONTAP y no es compatible con ONTAPI (ZAPI).

Prepárate para configurar el backend con controladores SAN de ONTAP

Entiende los requisitos y las opciones de autenticación para configurar un backend de ONTAP con controladores SAN de ONTAP.

Requisitos

Para todos los backends de ONTAP, Trident requiere que al menos un agregado esté asignado a la SVM.

> "Sistemas ASA r2" se diferencian de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. En los sistemas ASA r2, se utilizan zonas de disponibilidad de almacenamiento en lugar de agregados. Consulta el "esto" artículo de base de conocimientos sobre cómo asignar agregados a SVMs en sistemas ASA r2.

Recuerda que también puedes ejecutar más de un controlador y crear clases de almacenamiento que apunten a uno u otro. Por ejemplo, podrías configurar una `san-dev` clase que use el `ontap-san` controlador y una `san-default` clase que use el `ontap-san-economy` otro.

Todos tus nodos de trabajo de Kubernetes deben tener instaladas las herramientas iSCSI adecuadas. Consulta "Prepara el nodo trabajador" para más detalles.

Autentica el backend de ONTAP

Trident ofrece dos modos de autenticación para un backend ONTAP.

- Basado en credenciales: el nombre de usuario y la contraseña de un usuario de ONTAP con los permisos necesarios. Se recomienda usar un rol de inicio de sesión de seguridad predefinido, como `admin` o `vsadmin` para garantizar la máxima compatibilidad con las versiones de ONTAP.

- Basado en certificado: Trident también puede comunicarse con un clúster de ONTAP usando un certificado instalado en el backend. Aquí, la definición del backend debe contener valores codificados en Base64 del certificado del cliente, la clave y el certificado de la CA de confianza si se usa (recomendado).

Puedes actualizar los backends existentes para pasar de métodos basados en credenciales a métodos basados en certificados y viceversa. Sin embargo, solo se admite un método de autenticación a la vez. Para cambiar a otro método de autenticación, debes eliminar el método existente de la configuración del backend.

Si intentas proporcionar **tanto credenciales como certificados**, la creación del backend fallará con un error que indica que se proporcionó más de un método de autenticación en el archivo de configuración.

Habilita la autenticación basada en credenciales

Trident requiere las credenciales de un administrador con ámbito de SVM o de clúster para comunicarse con el backend de ONTAP. Se recomienda usar roles estándar predefinidos como `admin` o `vsadmin`. Esto garantiza la compatibilidad futura con versiones de ONTAP que podrían exponer APIs de funciones para futuras versiones de Trident. Se puede crear y usar un rol personalizado de inicio de sesión de seguridad con Trident, pero no se recomienda.

Un ejemplo de definición de backend se verá así:

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"
}
```

Ten en cuenta que la definición del backend es el único lugar donde se almacenan las credenciales en texto plano. Después de crear el backend, los nombres de usuario y contraseñas se codifican con Base64 y se

guardan como secretos de Kubernetes. La creación o actualización de un backend es el único paso que requiere conocimiento de las credenciales. Así que es una operación solo para el administrador, que debe realizar el administrador de Kubernetes o de almacenamiento.

Habilita la autenticación basada en certificados

Los backends, tanto nuevos como existentes, pueden usar un certificado y comunicarse con el backend de ONTAP. Se requieren tres parámetros en la definición del backend.

- clientCertificate: valor codificado en Base64 del certificado de cliente.
- clientPrivateKey: Valor codificado en Base64 de la clave privada asociada.
- trustedCACertificate: Valor codificado en Base64 del certificado de CA de confianza. Si usas una CA de confianza, tienes que proporcionar este parámetro. Esto se puede ignorar si no usas una CA de confianza.

Un flujo de trabajo típico implica los siguientes pasos.

Pasos

1. Genera un certificado y una clave de cliente. Al generarlos, asigna el nombre común (CN) al usuario de ONTAP con el que te vas a autenticar.

```
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. Agrega un certificado de CA de confianza al clúster de ONTAP. Esto puede que ya lo haya gestionado el administrador de almacenamiento. Ignora esto si no se usa ninguna CA de confianza.

```
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. Instala el certificado y la clave del cliente (del paso 1) en el clúster ONTAP.

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

> Después de ejecutar este comando, ONTAP solicita la introducción del certificado. Pega el contenido del archivo `k8senv.pem` generado en el paso 1, luego ingresa `END` para completar la instalación.

4. Confirma que el rol de inicio de sesión de seguridad de ONTAP admite `cert` el método de autenticación.

```
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. Prueba la autenticación usando el certificado generado. Reemplaza <ONTAP Management LIF> y <vserver name> con la IP de la LIF de administración y el nombre de la SVM.

```
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. Codifica el certificado, la clave y el certificado de CA confiable con Base64.

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

7. Crea un backend usando los valores obtenidos en el paso anterior.

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

Actualiza los métodos de autenticación o rota las credenciales

Puedes actualizar un backend existente para usar un método de autenticación diferente o para rotar sus credenciales. Esto funciona en ambos sentidos: los backends que usan nombre de usuario y contraseña pueden actualizarse para usar certificados; los backends que utilizan certificados pueden actualizarse para usar nombre de usuario y contraseña. Para hacer esto, debes eliminar el método de autenticación existente y agregar el nuevo método de autenticación. Luego usa el archivo backend.json actualizado que contiene los parámetros necesarios para ejecutar `tridentctl backend update`.

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

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

Al rotar contraseñas, el administrador de almacenamiento debe primero actualizar la contraseña del usuario en ONTAP. Luego, se realiza una actualización del backend. Al rotar certificados, se pueden agregar varios certificados al usuario. Después, el backend se actualiza para usar el nuevo certificado, y luego se puede eliminar el certificado antiguo del clúster de ONTAP.

Actualizar un backend no interrumpe el acceso a los volúmenes ya creados ni afecta las conexiones de volumen realizadas después. Una actualización correcta del backend indica que Trident puede comunicarse con el backend de ONTAP y gestionar futuras operaciones de volumen.

Crear rol ONTAP personalizado para Trident

Puedes crear un rol de clúster de ONTAP con privilegios mínimos para que no tengas que usar el rol de admin de ONTAP para realizar operaciones en Trident. Cuando incluyes el nombre de usuario en una configuración de backend de Trident, Trident usa el rol de clúster de ONTAP que creaste para realizar las operaciones.

Consulta "Generador de roles personalizados de Trident" para más información sobre cómo crear roles personalizados de Trident.

Usando ONTAP CLI

1. Crea un nuevo rol usando el siguiente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea un nombre de usuario para el usuario Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Asigna el rol al usuario:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Realiza los siguientes pasos en ONTAP System Manager:

1. **Crea un rol personalizado**:
   a. Para crear un rol personalizado a nivel de cluster, selecciona **Cluster > Settings**.

      (O) Para crear un rol personalizado a nivel de SVM, selecciona **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecciona el icono de flecha (→) junto a **Users and Roles**.
   c. Selecciona **+Add** en **Roles**.
   d. Define las reglas para el rol y haz clic en **Guardar**.
2. **Asigna el rol al usuario Trident**: + Realiza los siguientes pasos en la página **Usuarios y roles**:
   a. Selecciona el icono Add **+** en **Usuarios**.
   b. Selecciona el nombre de usuario requerido y elige un rol en el menú desplegable de **Role**.
   c. Haz clic en **Guardar**.

Consulta las siguientes páginas para obtener más información:

- "Roles personalizados para la administración de ONTAP" o "Define roles personalizados"
- "Trabaja con roles y usuarios"

Autentica conexiones con CHAP bidireccional

Trident puede autenticar sesiones iSCSI con CHAP bidireccional para los `ontap-san` y `ontap-san-economy` controladores. Esto requiere habilitar la opción `useCHAP` en la definición de tu backend. Cuando se establece en `true`, Trident configura la seguridad del iniciador predeterminado de la SVM como CHAP bidireccional y establece el nombre de usuario y los secretos desde el archivo del backend. NetApp recomienda usar CHAP bidireccional para autenticar conexiones. Mira la siguiente configuración de ejemplo:

```
---
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
```

El `useCHAP`parámetro es una opción booleana que solo se puede configurar una vez. Se establece en falso de forma predeterminada. Después de que lo configuras en verdadero, ya no puedes ponerlo en falso.

Además de `useCHAP=true`, los `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` y `chapUsername` campos deben incluirse en la definición del backend. Los secretos se pueden cambiar después de crear un backend ejecutando `tridentctl update`.

Cómo funciona

Al establecer `useCHAP` en true, el administrador de almacenamiento le indica a Trident que configure CHAP en el backend de almacenamiento. Esto incluye lo siguiente:

- Configurar CHAP en la SVM:
  - Si el tipo de seguridad del iniciador predeterminado de la SVM es ninguno (establecido de manera predeterminada) **y** no hay LUN preexistentes en el volumen, Trident establecerá el tipo de seguridad predeterminado en `CHAP` y procederá a configurar el nombre de usuario y los secretos del iniciador y del destino CHAP.
  - Si la SVM contiene LUNs, Trident no habilitará CHAP en la SVM. Esto garantiza que el acceso a los LUNs que ya están presentes en la SVM no esté restringido.
- Configurar el iniciador CHAP y el nombre de usuario y los secretos del destino; estas opciones deben especificarse en la configuración del backend (como se muestra arriba).

Después de que se crea el backend, Trident crea un CRD correspondiente `tridentbackend` y almacena los secretos y nombres de usuario CHAP como secretos de Kubernetes. Todos los PV que Trident cree en este backend se montarán y conectarán mediante CHAP.

Rotar credenciales y actualizar backends

Puedes actualizar las credenciales CHAP actualizando los parámetros CHAP en el archivo `backend.json`. Esto requerirá actualizar los secretos CHAP y usar el comando `tridentctl update` para reflejar estos cambios.

Al actualizar los secretos CHAP de un backend, debes usar `tridentctl` para actualizar el backend. No actualices las credenciales en el clúster de almacenamiento usando la CLI de ONTAP o el System Manager de ONTAP, ya que Trident no podrá detectar estos cambios.

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

Las conexiones existentes no se verán afectadas; seguirán activas si Trident actualiza las credenciales en la SVM. Las nuevas conexiones usan las credenciales actualizadas y las conexiones existentes siguen activas. Desconectar y volver a conectar los PV antiguos hará que usen las credenciales actualizadas.

Opciones de configuración y ejemplos de ONTAP SAN

Aprende cómo crear y usar controladores ONTAP SAN con tu instalación de Trident. Esta sección ofrece ejemplos de configuración de backends y detalles para asignar backends a StorageClasses.

"Sistemas ASA r2" difieren de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento. Estas variaciones afectan el uso de ciertos parámetros como se indica. "Conoce más sobre las diferencias entre los sistemas ASA r2 y otros sistemas ONTAP".

Solo el controlador `ontap-san` (con protocolos iSCSI, NVMe/TCP y FC) es compatible con los sistemas ASA r2.

En la configuración del backend de Trident, no necesitas especificar que tu sistema es ASA r2. Cuando seleccionas `ontap-san` como `storageDriverName`, Trident detecta automáticamente el ASA r2 u otros sistemas ONTAP. Algunos parámetros de configuración del backend no aplican a los sistemas ASA r2, como se indica en la tabla de abajo.

Opciones de configuración del backend

Consulta la siguiente tabla para ver las opciones de configuración del backend:

Parámetro	Descripción	Predeterminado
`version`		Siempre 1
`storageDriverName`	Nombre del controlador de almacenamiento	`ontap-san` o `ontap-san-economy`
`backendName`	Nombre personalizado o el backend de almacenamiento	Nombre del driver + "_" + dataLIF
`managementLIF`	Dirección IP de un LIF de gestión de clúster o SVM. Se puede especificar un nombre de dominio completo (FQDN). Puede configurarse para usar direcciones IPv6 si Trident se instaló usando la bandera IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para una conmutación de sitios sin interrupciones de MetroCluster, consulta el Ejemplo de MetroCluster. Si usas credenciales "vsadmin", `managementLIF` debe ser el de la SVM; si usas credenciales "admin", `managementLIF` debe ser el del clúster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Dirección IP del LIF de protocolo. Puede configurarse para usar direcciones IPv6 si Trident se instaló usando el indicador IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **No especifiques para iSCSI.** Trident usa "Mapa LUN selectivo de ONTAP" para descubrir los LIF de iSCSI necesarios para establecer una sesión multipath. Se genera una advertencia si `dataLIF` se define explícitamente. **Omite para MetroCluster.** Consulta la Ejemplo de MetroCluster.	Derivado por la SVM

Parámetro	Descripción	Predeterminado
`svm`	Máquina virtual de almacenamiento a usar **Omitir para MetroCluster.** Consulta el Ejemplo de MetroCluster.	Se deriva si se especifica un SVM `managementLIF`
`useCHAP`	Usa CHAP para autenticar iSCSI para los controladores ONTAP SAN [parámetro booleano]. Establécelo en `true` para que Trident configure y use CHAP bidireccional como la autenticación predeterminada para el SVM dado en el backend. Consulta "Prepárate para configurar el backend con controladores SAN de ONTAP" para más detalles. **No es compatible con FCP o NVMe/TCP.**	`false`
`chapInitiatorSecret`	Secreto del iniciador de CHAP. Obligatorio si `useCHAP=true`	""
`labels`	Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes	""
`chapTargetInitiatorSecret`	Secreto del iniciador de destino CHAP. Obligatorio si `useCHAP=true`	""
`chapUsername`	Nombre de usuario de entrada. Obligatorio si `useCHAP=true`	""
`chapTargetUsername`	Nombre de usuario de destino. Requerido si `useCHAP=true`	""
`clientCertificate`	Valor codificado en Base64 del certificado del cliente. Usado para autenticación basada en certificados	""
`clientPrivateKey`	Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados	""
`trustedCACertificate`	Valor codificado en Base64 del certificado de CA de confianza. Opcional. Se usa para la autenticación basada en certificados.	""
`username`	Nombre de usuario necesario para comunicarte con el clúster ONTAP. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, consulta "Autentica Trident en un SVM backend usando credenciales de Active Directory".	""
`password`	Contraseña necesaria para comunicarte con el clúster ONTAP. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, mira "Autentica Trident en un SVM backend usando credenciales de Active Directory".	""
`svm`	Máquina virtual de almacenamiento que vas a usar	Se deriva si se especifica un SVM `managementLIF`

Parámetro	Descripción	Predeterminado
`storagePrefix`	Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede modificar después. Para actualizar este parámetro, tendrás que crear un nuevo backend.	`trident`
`aggregate`	Agregado para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Para el `ontap-nas-flexgroup` driver, esta opción se ignora. Si no se asigna, cualquiera de los agregados disponibles se puede usar para aprovisionar un FlexGroup volumen. Cuando el agregado se actualiza en SVM, se actualiza automáticamente en Trident mediante el sondeo de SVM sin tener que reiniciar el Trident Controller. Cuando has configurado un agregado específico en Trident para aprovisionar volúmenes, si el agregado se renombra o se mueve fuera de la SVM, el backend pasará a estado fallido en Trident mientras sondea el agregado de la SVM. Debes cambiar el agregado por uno que esté presente en la SVM o eliminarlo por completo para que el backend vuelva a estar en línea. **No especifiques para sistemas ASA r2**.	""
`limitAggregateUsage`	Falla el aprovisionamiento si el uso es superior a este porcentaje. Si estás usando un backend de Amazon FSx para NetApp ONTAP, no especifiques `limitAggregateUsage`. Los `fsxadmin` proporcionados y `vsadmin` no contienen los permisos necesarios para recuperar el uso agregado y limitarlo usando Trident. **No especifiques para sistemas ASA r2**.	"" (no aplicado por defecto)
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor. También restringe el tamaño máximo de los volúmenes que gestiona para LUNs.	"" (no aplicado por defecto)
`lunsPerFlexvol`	Máximo de LUNs por FlexVol, debe estar en el rango [50, 200]	`100`
`debugTraceFlags`	Indicadores de depuración para utilizar cuando estés solucionando problemas. Por ejemplo, {"api":false, "method":true} no lo uses a menos que estés solucionando problemas y necesites un volcado detallado del registro.	`null`

Parámetro	Descripción	Predeterminado
`useREST`	Parámetro booleano para usar las API REST de ONTAP. `useREST`Cuando se configura en `true`, Trident usa las API REST de ONTAP para comunicarse con el backend; cuando se configura en `false`, Trident usa llamadas ONTAPI (ZAPI) para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontapi`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`. A partir de la versión Trident 24.06 y ONTAP 9.15.1 o posteriores, `useREST` se configura en `true` de forma predeterminada; cambia `useREST` a `false` para usar llamadas ONTAPI (ZAPI). `useREST` está totalmente calificado para NVMe/TCP. NVMe solo es compatible con las API REST de ONTAP y no es compatible con ONTAPI (ZAPI). **Si se especifica, siempre configúralo en `true` para sistemas ASA r2**.	`true` para ONTAP 9.15.1 o posterior, de lo contrario `false`.
`sanType`	Usa para seleccionar `iscsi` para iSCSI, `nvme` para NVMe/TCP o `fcp` para SCSI sobre Fibre Channel (FC).	`iscsi` si está en blanco

Parámetro	Descripción	Predeterminado
`formatOptions`	Usa `formatOptions` para especificar argumentos de la línea de comandos para el comando `mkfs`, que se aplicarán cada vez que se formatee un volumen. Esto te permite formatear el volumen según tus preferencias. Asegúrate de especificar formatOptions similar a las opciones del comando mkfs, excluyendo la ruta del dispositivo. Ejemplo: "-E nodiscard" **Compatible con `ontap-san` y `ontap-san-economy` controladores con el protocolo iSCSI. Además, compatible con sistemas ASA r2 cuando usas los protocolos iSCSI y NVMe/TCP.**	
`limitVolumePoolSize`	Tamaño máximo solicitable de FlexVol al utilizar LUNs en el backend de ontap-san-economy.	"" (no aplicado por defecto)
`denyNewVolumePools`	Restringe `ontap-san-economy` a los backends crear nuevos volúmenes FlexVol para contener sus LUN. Solo se usan FlexVols preexistentes para aprovisionar nuevos PV.	

Recomendaciones para usar formatOptions

Trident recomienda las siguientes opciones para acelerar el proceso de formateo:

- **-E nodiscard (ext3, ext4):** No intentes descartar bloques al momento de ejecutar mkfs (descartar bloques inicialmente es útil en dispositivos de estado sólido y almacenamiento disperso o Thin-Provisioning). Esto reemplaza la opción obsoleta "-K" y es aplicable a los sistemas de archivos ext3 y ext4.
- **-K (xfs):** No intentes descartar bloques durante la ejecución de mkfs. Esta opción es aplicable al sistema de archivos xfs.

Autentica Trident en un SVM backend usando credenciales de Active Directory

Puedes configurar Trident para que se autentique en una SVM de backend usando credenciales de Active Directory (AD). Antes de que una cuenta de AD pueda acceder a la SVM, tienes que configurar el acceso del controlador de dominio de AD al clúster o a la SVM. Para la administración del clúster con una cuenta de AD, tienes que crear un domain tunnel. Consulta "Configura el acceso al controlador de dominio de Active Directory en ONTAP" para más detalles.

pasos

1. Configura los ajustes de Domain Name System (DNS) para un SVM de backend:

   ```
   vserver services dns create -vserver <svm_name> -dns-servers
   <dns_server_ip1>,<dns_server_ip2>
   ```

2. Ejecuta el siguiente comando para crear una cuenta de equipo para la SVM en Active Directory:

   ```
   vserver active-directory create -vserver DataSVM -account-name ADSERVER1
   -domain demo.netapp.com
   ```

3. Usa este comando para crear un usuario o grupo de AD para administrar el clúster o SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. En el archivo de configuración del backend de Trident, establece los parámetros `username` y `password` en el nombre de usuario o grupo de AD y la contraseña, respectivamente.

Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

Parámetro	Descripción	Predeterminado
`spaceAllocation`	Asignación de espacio para LUNs	"verdadero" **Si se especifica, configúralo en `true` para sistemas ASA r2**.
`spaceReserve`	Modo de reserva de espacio; "ninguno" (thin) o "volumen" (thick). **Configúralo en `none` sistemas ASA r2**.	"none"
`snapshotPolicy`	Política de SnapVault que se va a usar. **Establece en `none` para sistemas ASA r2**.	"none"
`qosPolicy`	Grupo de políticas de QoS para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento/backend. Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o una versión posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite para el rendimiento total de todas las cargas de trabajo.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por cada pool de almacenamiento/backend	""
`snapshotReserve`	Porcentaje de volumen reservado para instantáneas. **No especificar para sistemas ASA r2**.	"0" si `snapshotPolicy` es "none", de lo contrario ""
`splitOnClone`	Divide un clon de su padre al momento de su creación	"false"
`encryption`	Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE".	"falso" **Si se especifica, configúralo en `true` para sistemas ASA r2**.

Parámetro	Descripción	Predeterminado
`luksEncryption`	Habilita el cifrado LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)".	"" **Establécelo en `false` para sistemas ASA r2.**
`tieringPolicy`	Política de niveles para usar "none" **No especificar para sistemas ASA r2.**	
`nameTemplate`	Plantilla para crear nombres de volúmenes personalizados.	""

Ejemplos de aprovisionamiento de volumen

Aquí tienes un ejemplo con valores predeterminados definidos:

```
---
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'
```

> Para todos los volúmenes creados usando el `ontap-san` driver, Trident añade un 10 por ciento extra de capacidad al FlexVol para alojar los metadatos del LUN. El LUN se aprovisionará con el tamaño exacto que el usuario solicite en la PVC. Trident añade un 10 por ciento al FlexVol (se muestra como tamaño disponible en ONTAP). Ahora los usuarios obtendrán la cantidad de capacidad utilizable que solicitaron. Este cambio también evita que los LUN se vuelvan de solo lectura a menos que se utilice completamente el espacio disponible. Esto no aplica a ontap-san-economy.

Para los backends que definen `snapshotReserve`, Trident calcula el tamaño de los volúmenes de la siguiente manera:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

El 1.1 es el 10 % adicional que Trident añade a la FlexVol para alojar los metadatos del LUN. Para `snapshotReserve` = 5 %, y una solicitud de PVC de 5 GiB, el tamaño total del volumen es 5.79 GiB y el tamaño disponible es 5.5 GiB. El `volume show` comando debería mostrar resultados similares a este ejemplo:

Actualmente, cambiar el tamaño es la única manera de usar el nuevo cálculo para un volumen existente.

Ejemplos de configuración mínima

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

> Si estás usando Amazon FSx en NetApp ONTAP con Trident, NetApp recomienda que especifiques nombres DNS para las LIFs en vez de direcciones IP.

Ejemplo de ONTAP SAN

Esta es una configuración básica usando el `ontap-san` driver.

```
---
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>
```

Ejemplo de MetroCluster

Puedes configurar el backend para evitar tener que actualizar manualmente la definición del backend después de la conmutación de sitios y la conmutación de vuelta durante "Replicación y recuperación de SVM".

Para una conmutación de sitios y reversión sin problemas, especifica el SVM usando `managementLIF` y omite los parámetros `svm`. Por ejemplo:

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

Ejemplo de ONTAP SAN economy

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

Ejemplo de autenticación basada en certificados

En este ejemplo de configuración básica `clientCertificate`, `clientPrivateKey` y `trustedCACertificate` (opcional, si usas una CA confiable) se completan en `backend.json` y toman los valores codificados en base64 del certificado del cliente, la clave privada y el certificado de CA confiable, respectivamente.

```
---
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
```

Ejemplos de CHAP bidireccional

Estos ejemplos crean un backend con `useCHAP` configurado en `true`.

Ejemplo de ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Ejemplo de ONTAP SAN economy CHAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Ejemplo de NVMe/TCP

Debes tener una SVM configurada con NVMe en tu backend de ONTAP. Esta es una configuración básica de backend para 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
```

Ejemplo de SCSI sobre FC (FCP)

Debes tener una SVM configurada con FC en tu backend de ONTAP. Esta es una configuración básica de backend para 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
```

Ejemplo de configuración de backend con 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 ejemplo para el controlador ontap-san-economy

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Ejemplos de backends con pools virtuales

En estos archivos de definición de backend de ejemplo, se establecen valores predeterminados específicos para todos los grupos de almacenamiento, como `spaceReserve` en ninguno, `spaceAllocation` en falso y `encryption` en falso. Los grupos virtuales se definen en la sección de almacenamiento.

Trident establece las etiquetas de aprovisionamiento en el campo "Comentarios". Los comentarios se establecen en el volumen FlexVol. Trident copia todas las etiquetas presentes en un pool virtual al volumen de almacenamiento durante el aprovisionamiento. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En estos ejemplos, algunos de los pools de almacenamiento establecen sus propios `spaceReserve`, `spaceAllocation` y `encryption` valores, y algunos pools anulan los valores predeterminados.

Ejemplo de ONTAP SAN

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Ejemplo de ONTAP SAN economy

```
---
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"
```

Ejemplo de NVMe/TCP

```
---
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"
```

Asigna backends a StorageClasses

Las siguientes definiciones de StorageClass se refieren a Ejemplos de backends con pools virtuales. Usando el campo `parameters.selector`, cada StorageClass indica qué grupos virtuales pueden usarse para alojar un volumen. El volumen tendrá los aspectos definidos en el grupo virtual elegido.

- El `protection-gold` StorageClass se asignará al primer pool virtual en el `ontap-san` backend. Este es el único pool que ofrece protección de nivel oro.

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

- El `protection-not-gold` StorageClass se asignará al segundo y tercer pool virtual en `ontap-san` backend. Estos son los únicos pools que ofrecen un nivel de protección distinto al oro.

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

- El `app-mysqldb` StorageClass se asignará al tercer pool virtual en `ontap-san-economy` backend. Este es el único pool que ofrece configuración de pool de almacenamiento para la app tipo mysqldb.

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

- El `protection-silver-creditpoints-20k` StorageClass se asignará al segundo pool virtual en `ontap-san` backend. Este es el único pool que ofrece protección de nivel plata y 20000 puntos de crédito.

```
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"
```

- El `creditpoints-5k` StorageClass se asignará al tercer grupo virtual en el `ontap-san` backend y al cuarto grupo virtual en el `ontap-san-economy` backend. Estas son las únicas ofertas de grupos con 5000 puntos de crédito.

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

- El `my-test-app-sc` StorageClass se asignará al `testAPP` grupo virtual en el `ontap-san` driver con `sanType: nvme`. Este es el único grupo que ofrece `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 decidirá qué grupo virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

Controladores ONTAP NAS

Descripción general del controlador NAS de ONTAP

Conoce cómo configurar un backend de ONTAP con los controladores NAS de ONTAP y Cloud Volumes ONTAP.

Detalles del controlador ONTAP NAS

Trident proporciona los siguientes controladores de almacenamiento NAS para comunicarse con el clúster ONTAP. Los modos de acceso compatibles son: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`ontap-nas`	NFS SMB	Sistema de archivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Controlador	Protocolo	volumeMode	Modos de acceso admitidos	Sistemas de archivos compatibles
`ontap-nas-economy`	NFS SMB	Sistema de archivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-flexgroup`	NFS SMB	Sistema de archivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

- Usa `ontap-san-economy` solo si esperas que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP".
- Usa `ontap-nas-economy` solo si se espera que el recuento de uso de volúmenes persistentes sea mayor que "límites de volúmenes compatibles de ONTAP" y no se puede usar el controlador `ontap-san-economy`.
- No uses `ontap-nas-economy` si crees que vas a necesitar protección de datos, recuperación ante desastres o movilidad.
- NetApp no recomienda usar FlexVol autogrow en todos los controladores ONTAP, excepto ontap-san. Como solución alternativa, Trident admite el uso de snapshot reserve y escala los volúmenes FlexVol en consecuencia.

Permisos de usuario

Trident espera ejecutarse como administrador de ONTAP o SVM, normalmente usando el usuario cluster `admin` o un usuario SVM `vsadmin`, o un usuario con un nombre diferente que tenga el mismo rol.

Para implementaciones de Amazon FSx para NetApp ONTAP, Trident espera ejecutarse como administrador de ONTAP o SVM, usando el usuario de clúster `fsxadmin` o un `vsadmin` usuario de SVM, o un usuario con un nombre diferente que tenga el mismo rol. El `fsxadmin` usuario es un reemplazo limitado para el usuario administrador de clúster.

Si usas el parámetro `limitAggregateUsage`, se requieren permisos de administrador de clúster. Cuando usas Amazon FSx for NetApp ONTAP con Trident, el parámetro `limitAggregateUsage` no funcionará con las cuentas de usuario `vsadmin` y `fsxadmin`. La operación de configuración fallará si especificas este parámetro.

Aunque es posible crear un rol más restrictivo dentro de ONTAP que un controlador Trident pueda usar, no lo recomendamos. La mayoría de las nuevas versiones de Trident llamarán a APIs adicionales que habría que tener en cuenta, lo que hace que las actualizaciones sean difíciles y propensas a errores.

Prepárate para configurar un backend con controladores ONTAP NAS

Entiende los requisitos, las opciones de autenticación y las políticas de exportación para configurar un backend ONTAP con drivers ONTAP NAS.

A partir de la versión 25.10, NetApp Trident es compatible con "Sistema de almacenamiento NetApp AFX". NetApp AFX storage systems difieren de otros sistemas ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento.

Sólo el `ontap-nas` controlador (con protocolo NFS) es compatible con los sistemas AFX; el protocolo SMB no es compatible.

En la configuración del backend de Trident, no necesitas especificar que tu sistema es AFX. Cuando seleccionas `ontap-nas` como `storageDriverName`, Trident detecta automáticamente los sistemas AFX.

Requisitos

- Para todos los backends de ONTAP, Trident requiere que al menos un agregado esté asignado a la SVM.
- Puedes ejecutar más de un controlador y crear clases de almacenamiento que apunten a uno u otro. Por ejemplo, podrías configurar una clase Gold que use el `ontap-nas` controlador y una clase Bronze que use el `ontap-nas-economy` otro.
- Todos tus nodos worker de Kubernetes deben tener instaladas las herramientas NFS adecuadas. Consulta "aquí" para más detalles.
- Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows. Consulta Prepárate para aprovisionar volúmenes SMB para más detalles.

Autentica el backend de ONTAP

Trident ofrece dos modos de autenticación para un backend ONTAP.

- Basado en credenciales: este modo requiere permisos suficientes en el backend de ONTAP. Se recomienda usar una cuenta asociada a un rol de inicio de sesión de seguridad predefinido, como `admin` o `vsadmin` para garantizar la máxima compatibilidad con las versiones de ONTAP.
- Basado en certificado: este modo requiere un certificado instalado en el backend para que Trident se comunique con un clúster de ONTAP. Aquí, la definición del backend debe contener valores codificados en Base64 del certificado del cliente, la clave y el certificado de la CA de confianza si se utiliza (recomendado).

Puedes actualizar los backends existentes para pasar de métodos basados en credenciales a métodos basados en certificados y viceversa. Sin embargo, solo se admite un método de autenticación a la vez. Para cambiar a otro método de autenticación, debes eliminar el método existente de la configuración del backend.

Si intentas proporcionar **tanto credenciales como certificados**, la creación del backend fallará con un error que indica que se proporcionó más de un método de autenticación en el archivo de configuración.

Habilita la autenticación basada en credenciales

Trident requiere las credenciales de un administrador con ámbito de SVM o de clúster para comunicarse con el backend de ONTAP. Se recomienda usar roles estándar predefinidos como `admin` o `vsadmin`. Esto garantiza la compatibilidad futura con versiones de ONTAP que podrían exponer APIs de funciones para futuras versiones de Trident. Se puede crear y usar un rol personalizado de inicio de sesión de seguridad con Trident, pero no se recomienda.

Un ejemplo de definición de backend se verá así:

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"
    }
}
```

Ten en cuenta que la definición del backend es el único lugar donde se almacenan las credenciales en texto plano. Después de crear el backend, los nombres de usuario y contraseñas se codifican con Base64 y se almacenan como secretos de Kubernetes. La creación o actualización de un backend es el único paso que requiere conocimiento de las credenciales. Así que es una operación solo para el administrador, que debe realizar el administrador de Kubernetes o el administrador de almacenamiento.

Habilita la autenticación basada en certificados

Los backends, tanto nuevos como existentes, pueden usar un certificado y comunicarse con el backend de ONTAP. Se requieren tres parámetros en la definición del backend.

- clientCertificate: valor codificado en Base64 del certificado de cliente.
- clientPrivateKey: Valor codificado en Base64 de la clave privada asociada.
- trustedCACertificate: Valor codificado en Base64 del certificado de CA de confianza. Si usas una CA de confianza, tienes que proporcionar este parámetro. Esto se puede ignorar si no usas una CA de confianza.

Un flujo de trabajo típico implica los siguientes pasos.

Pasos

1. Genera un certificado y una clave de cliente. Al generarlos, asigna el nombre común (CN) al usuario de

ONTAP con el que te vas a autenticar.

```
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. Agrega un certificado de CA de confianza al clúster de ONTAP. Esto puede que ya lo haya gestionado el administrador de almacenamiento. Ignora esto si no se usa ninguna CA de confianza.

```
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. Instala el certificado y la clave del cliente (del paso 1) en el clúster ONTAP.

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

4. Confirma que el rol de inicio de sesión de seguridad de ONTAP admite `cert` el método de autenticación.

```
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. Prueba la autenticación usando el certificado generado. Reemplaza <ONTAP Management LIF> y <vserver name> con la dirección IP de Management LIF y el nombre de SVM. Debes asegurarte de que el LIF tenga su política de servicio configurada en `default-data-management`.

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

6. Codifica el certificado, la clave y el certificado de CA confiable con Base64.

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

7. Crea un backend usando los valores obtenidos en el paso anterior.

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

Actualiza los métodos de autenticación o rota las credenciales

Puedes actualizar un backend existente para usar un método de autenticación diferente o para rotar sus credenciales. Esto funciona en ambos sentidos: los backends que usan nombre de usuario y contraseña pueden actualizarse para usar certificados; los backends que utilizan certificados pueden actualizarse para usar nombre de usuario y contraseña. Para hacer esto, debes eliminar el método de autenticación existente y agregar el nuevo método de autenticación. Luego usa el archivo backend.json actualizado que contiene los parámetros necesarios para ejecutar `tridentctl update backend`.

```
cat cert-backend-updated.json
```

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

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

Al rotar contraseñas, el administrador de almacenamiento debe primero actualizar la contraseña del usuario en ONTAP. Luego, se realiza una actualización del backend. Al rotar certificados, se pueden agregar varios certificados al usuario. Después, el backend se actualiza para usar el nuevo certificado, y luego se puede eliminar el certificado antiguo del clúster de ONTAP.

Actualizar un backend no interrumpe el acceso a los volúmenes ya creados ni afecta las conexiones de volumen realizadas después. Una actualización correcta del backend indica que Trident puede comunicarse con el backend de ONTAP y gestionar futuras operaciones de volumen.

Crear rol ONTAP personalizado para Trident

Puedes crear un rol de clúster de ONTAP con privilegios mínimos para que no tengas que usar el rol de admin de ONTAP para realizar operaciones en Trident. Cuando incluyes el nombre de usuario en una configuración de backend de Trident, Trident usa el rol de clúster de ONTAP que creaste para realizar las operaciones.

Consulta "Generador de roles personalizados de Trident" para más información sobre cómo crear roles personalizados de Trident.

Usando ONTAP CLI

1. Crea un nuevo rol usando el siguiente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crea un nombre de usuario para el usuario Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Asigna el rol al usuario:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Realiza los siguientes pasos en ONTAP System Manager:

1. **Crea un rol personalizado**:
   a. Para crear un rol personalizado a nivel de cluster, selecciona **Cluster > Settings**.

      (O) Para crear un rol personalizado a nivel de SVM, selecciona **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecciona el icono de flecha (→) junto a **Users and Roles**.
   c. Selecciona **+Add** en **Roles**.
   d. Define las reglas para el rol y haz clic en **Guardar**.
2. **Asigna el rol al usuario Trident**: + Realiza los siguientes pasos en la página **Usuarios y roles**:
   a. Selecciona el icono Add **+** en **Usuarios**.
   b. Selecciona el nombre de usuario requerido y elige un rol en el menú desplegable de **Role**.
   c. Haz clic en **Guardar**.

Consulta las siguientes páginas para obtener más información:

- "Roles personalizados para la administración de ONTAP" o "Define roles personalizados"
- "Trabaja con roles y usuarios"

Gestiona las políticas de exportación de NFS

Trident utiliza políticas de exportación NFS para controlar el acceso a los volúmenes que aprovisiona.

Trident ofrece dos opciones cuando trabajas con políticas de exportación:

- Trident puede gestionar dinámicamente la política de exportación por sí mismo; en este modo de funcionamiento, el administrador de almacenamiento especifica una lista de bloques CIDR que

representan direcciones IP admisibles. Trident añade automáticamente a la política de exportación las direcciones IP de nodo aplicables que estén dentro de estos rangos en el momento de la publicación. Como alternativa, cuando no se especifican CIDR, se añadirán a la política de exportación todas las direcciones IP de unidifusión de ámbito global que se encuentren en el nodo al que se está publicando el volumen.

- Los administradores de almacenamiento pueden crear una política de exportación y agregar reglas manualmente. Trident utiliza la política de exportación predeterminada a menos que se especifique un nombre de política de exportación diferente en la configuración.

Gestiona dinámicamente las políticas de exportación

Trident ofrece la capacidad de gestionar dinámicamente las políticas de exportación para los backends de ONTAP. Esto le da al administrador de almacenamiento la posibilidad de especificar un espacio de direcciones permitido para las direcciones IP de los nodos de trabajo, en vez de definir reglas explícitas manualmente. Esto simplifica mucho la gestión de las políticas de exportación; las modificaciones en la política de exportación ya no requieren intervención manual en el clúster de almacenamiento. Además, esto ayuda a restringir el acceso al clúster de almacenamiento solo a los nodos de trabajo que están montando volúmenes y tienen direcciones IP dentro del rango especificado, lo que permite una gestión automatizada y detallada.

> No uses Network Address Translation (NAT) cuando uses políticas de exportación dinámicas. Con NAT, el controlador de almacenamiento ve la dirección NAT del frontend y no la dirección IP real del host, así que se denegará el acceso cuando no se encuentre coincidencia en las reglas de exportación.

Ejemplo

Hay dos opciones de configuración que deben utilizarse. Aquí tienes un ejemplo de definición de backend:

```
---
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
```

> Al usar esta función, debes asegurarte de que la unión raíz de tu SVM tenga una política de exportación previamente creada con una regla de exportación que permita el bloque CIDR del nodo (como la política de exportación predeterminada). Sigue siempre la práctica recomendada por NetApp de dedicar una SVM para Trident.

Aquí tienes una explicación de cómo funciona esta función usando el ejemplo de arriba:

- `autoExportPolicy` se establece en `true`. Esto indica que Trident crea una política de exportación para cada volumen aprovisionado con este backend para la `svm1` SVM y gestiona la adición y eliminación de

reglas usando bloques de direcciones `autoexportCIDRs`. Hasta que un volumen se conecta a un nodo, el volumen utiliza una política de exportación vacía sin reglas para evitar el acceso no deseado a ese volumen. Cuando un volumen se publica en un nodo, Trident crea una política de exportación con el mismo nombre que el qtree subyacente que contiene la dirección IP del nodo dentro del bloque CIDR especificado. Estas direcciones IP también se añadirán a la política de exportación utilizada por el volumen principal FlexVol.

  - Por ejemplo:
    - UUID de backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` establecer en `true`
    - prefijo de almacenamiento `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - El qtree llamado trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crea una política de exportación para el FlexVol llamado `trident-403b5326-8482-40db96d0-d83fb3f4daec`, una política de exportación para el qtree llamado `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, y una política de exportación vacía llamada `trident_empty` en la SVM. Las reglas para la política de exportación del FlexVol serán un superconjunto de cualquier regla contenida en las políticas de exportación del qtree. La política de exportación vacía se reutilizará para cualquier volumen que no esté adjunto.

- `autoExportCIDRs` contiene una lista de bloques de direcciones. Este campo es opcional y su valor predeterminado es ["0.0.0.0/0", "::/0"]. Si no se define, Trident agrega todas las direcciones unicast de alcance global que se encuentran en los nodos de trabajo con publicaciones.

En este ejemplo, el espacio de direcciones `192.168.0.0/24` se proporciona. Esto indica que las direcciones IP de los nodos de Kubernetes que estén dentro de este rango de direcciones con publicaciones se añadirán a la política de exportación que crea Trident. Cuando Trident registra un nodo en el que se ejecuta, recupera las direcciones IP del nodo y las compara con los bloques de direcciones proporcionados en `autoExportCIDRs`. Al momento de publicar, después de filtrar las IP, Trident crea las reglas de la política de exportación para las direcciones IP de cliente del nodo al que está publicando.

Puedes actualizar `autoExportPolicy` y `autoExportCIDRs` para los backends después de crearlos. Puedes añadir nuevos CIDR para un backend que se gestiona automáticamente o eliminar los CIDR existentes. Ten cuidado al eliminar los CIDR para asegurarte de que no se interrumpan las conexiones existentes. También puedes elegir deshabilitar `autoExportPolicy` para un backend y volver a una política de exportación creada manualmente. Esto requerirá configurar el parámetro `exportPolicy` en la configuración del backend.

Después de que Trident crea o actualiza un backend, puedes verificar el backend usando `tridentctl` o el correspondiente `tridentbackend` CRD:

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

Al eliminar un nodo, Trident revisa todas las políticas de exportación para eliminar las reglas de acceso correspondientes al nodo. Al eliminar esta dirección IP de nodo de las políticas de exportación de los backends administrados, Trident evita montajes no autorizados, a menos que esta IP sea reutilizada por un nuevo nodo en el clúster.

Para los backends preexistentes, actualizar el backend con `tridentctl update backend` asegura que Trident gestione las políticas de exportación automáticamente. Esto crea dos nuevas políticas de exportación nombradas según el UUID y el nombre del qtree del backend cuando sean necesarias. Los volúmenes que están presentes en el backend usarán las nuevas políticas de exportación después de desmontarlos y volver a montarlos.

Al eliminar un backend con políticas de exportación autogestionadas, se eliminará la política de exportación creada dinámicamente. Si el backend se vuelve a crear, se trata como un backend nuevo y resultará en la creación de una nueva política de exportación.

Si se actualiza la dirección IP de un nodo activo, debes reiniciar el pod de Trident en el nodo. Trident actualizará la política de exportación para los backends que administra para reflejar este cambio de IP.

Prepárate para aprovisionar volúmenes SMB

Con un poco de preparación adicional, puedes aprovisionar volúmenes SMB usando `ontap-nas`drivers.

Debes configurar tanto los protocolos NFS como SMB/CIFS en la SVM para crear un `ontap-nas-economy` volumen SMB para clústeres ONTAP locales. Si no configuras alguno de estos protocolos, la creación del volumen SMB fallará.

`autoExportPolicy` no es compatible con volúmenes SMB.

Antes de empezar

Antes de que puedas aprovisionar volúmenes SMB, debes tener lo siguiente.

- Un clúster de Kubernetes con un nodo controlador Linux y al menos un nodo trabajador Windows que ejecuta Windows Server 2022. Trident admite volúmenes SMB montados en pods que se ejecutan solo en nodos de Windows.
- Al menos un secreto de Trident con tus credenciales de Active Directory. Para generar un secreto `smbcreds`:

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

- Un proxy CSI configurado como un servicio de Windows. Para configurar un `csi-proxy`, consulta "GitHub: CSI Proxy" o "GitHub: CSI Proxy para Windows" para nodos de Kubernetes que se ejecutan en Windows.

Pasos

1. Para ONTAP local, puedes crear opcionalmente un recurso compartido SMB o Trident puede crear uno para ti.

   Se requieren recursos compartidos SMB para Amazon FSx for ONTAP.

   Puedes crear los recursos compartidos de administrador de SMB de dos maneras: usando el complemento "Microsoft Management Console" Shared Folders o usando la CLI de ONTAP. Para crear los recursos compartidos de SMB usando la CLI de ONTAP:

   a. Si es necesario, crea la estructura de ruta del directorio para el recurso compartido.

      El `vserver cifs share create`comando comprueba la ruta especificada en la opción -path durante la creación del recurso compartido. Si la ruta especificada no existe, el comando falla.

   b. Crea un recurso compartido SMB asociado con la SVM especificada:

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

   c. Verifica que se creó el recurso compartido:

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

      Consulta "Crear un recurso compartido SMB" para obtener detalles completos.

2. Al crear el backend, debes configurar lo siguiente para especificar los volúmenes SMB. Para todas las

opciones de configuración del backend de FSx for ONTAP, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

Parámetro	Descripción	Ejemplo
`smbShare`	Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco.	`smb-share`
`nasType`	**Debe establecerse en `smb`.** Si es nulo, el valor predeterminado es `nfs`.	`smb`
`securityStyle`	Estilo de seguridad para nuevos volúmenes. **Debe configurarse en `ntfs` o `mixed` para volúmenes SMB.**	`ntfs` o `mixed` para volúmenes SMB
`unixPermissions`	Modo para nuevos volúmenes. **Debe dejarse vacío para volúmenes SMB.**	""

Habilita SMB seguro

A partir de la versión 25.06, NetApp Trident admite el aprovisionamiento seguro de volúmenes SMB creados mediante `ontap-nas` y `ontap-nas-economy` backends. Cuando SMB seguro está habilitado, puedes proporcionar acceso controlado a los recursos compartidos SMB para usuarios y grupos de usuarios de Active Directory (AD) usando listas de control de acceso (ACL).

Puntos para recordar

- No se admite la importación `ontap-nas-economy` de volúmenes.
- Solo se admiten clones de solo lectura para `ontap-nas-economy` volúmenes.
- Si Secure SMB está habilitado, Trident ignorará el recurso compartido SMB mencionado en el backend.
- Actualizar la anotación de PVC, la anotación de la storage class y el campo backend no actualiza la ACL del recurso compartido SMB.
- La ACL de recurso compartido SMB especificada en la anotación del PVC clonado tendrá prioridad sobre las del PVC de origen.
- Asegúrate de proporcionar usuarios de AD válidos al habilitar SMB seguro. Los usuarios no válidos no se añadirán a la ACL.
- Si proporcionas el mismo usuario de AD en el backend, la clase de almacenamiento y el PVC con diferentes permisos, la prioridad de permisos será: PVC, clase de almacenamiento y luego backend.
- SMB seguro es compatible para `ontap-nas`importaciones de volúmenes administrados y no aplica a importaciones de volúmenes no administrados.

Pasos

1. Especifica adAdminUser en TridentBackendConfig como se muestra en el siguiente ejemplo:

```
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. Añade la anotación en la clase de almacenamiento.

   Agrega la `trident.netapp.io/smbShareAdUser` anotación a la clase de almacenamiento para habilitar SMB seguro sin fallar. El valor de usuario especificado para la anotación `trident.netapp.io/smbShareAdUser` debe ser el mismo que el nombre de usuario especificado en el `smbcreds` secreto. Puedes elegir una de las siguientes para `smbShareAdUserPermission`: `full_control`, `change` o `read`. El permiso predeterminado es `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. Crea un PVC.

   El siguiente ejemplo crea una 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
```

Opciones de configuración y ejemplos de ONTAP NAS

Aprende a crear y usar controladores NAS de ONTAP con tu instalación de Trident. Esta sección proporciona ejemplos de configuración de backend y detalles para asignar backends a StorageClasses.

A partir de la versión 25.10, NetApp Trident admite "NetApp sistemas de almacenamiento AFX". Los sistemas de almacenamiento NetApp AFX se diferencian de otros sistemas basados en ONTAP (ASA, AFF y FAS) en la implementación de su capa de almacenamiento.

> Sólo el `ontap-nas` controlador (con protocolo NFS) es compatible con NetApp AFX systems; el protocolo SMB no es compatible.

En la configuración del backend de Trident, no necesitas especificar que tu sistema es un sistema de almacenamiento NetApp AFX. Cuando seleccionas `ontap-nas` como el `storageDriverName`, Trident detecta automáticamente el sistema de almacenamiento AFX. Algunos parámetros de configuración del backend no aplican a los sistemas de almacenamiento AFX, como se indica en la tabla a continuación.

Opciones de configuración del backend

Consulta la siguiente tabla para ver las opciones de configuración del backend:

Parámetro	Descripción	Predeterminado
`version`		Siempre 1

Parámetro	Descripción	Predeterminado
`storageDriverName`	Nombre del controlador de almacenamiento Para los sistemas AFX de NetApp, solo `ontap-nas` es compatible.	`ontap-nas`, `ontap-nas-economy`, o `ontap-nas-flexgroup`
`backendName`	Nombre personalizado o el backend de almacenamiento	Nombre del driver + "_" + dataLIF
`managementLIF`	Dirección IP de un LIF de administración de clúster o SVM. Se puede especificar un nombre de dominio completo (FQDN). Se puede configurar para usar direcciones IPv6 si Trident se instaló con el indicador IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para una conmutación de sitios fluida de MetroCluster, consulta la Ejemplo de MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Dirección IP del protocolo LIF. NetApp recomienda especificar `dataLIF`. Si no se proporciona, Trident obtiene los dataLIFs de la SVM. Puedes especificar un nombre de dominio completo (FQDN) para las operaciones de montaje NFS, lo que te permite crear un DNS round-robin para equilibrar la carga entre varios dataLIFs. Se puede cambiar después de la configuración inicial. Consulta . Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el indicador IPv6. Las direcciones IPv6 deben definirse entre corchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omitir para MetroCluster.** Consulta Ejemplo de MetroCluster.	Dirección especificada o derivada de SVM, si no se especifica (no recomendado)
`svm`	Máquina virtual de almacenamiento a usar **Omitir para MetroCluster.** Consulta el Ejemplo de MetroCluster.	Se deriva si se especifica un SVM `managementLIF`
`autoExportPolicy`	Habilita la creación y actualización automática de políticas de exportación [Boolean]. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente.	false
`autoExportCIDRs`	Lista de CIDR para filtrar las direcciones IP de los nodos de Kubernetes cuando `autoExportPolicy` está habilitado. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente.	["0.0.0.0/0", "::/0"]`
`labels`	Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes	""

Parámetro	Descripción	Predeterminado
`clientCertificate`	Valor codificado en Base64 del certificado del cliente. Usado para autenticación basada en certificados	""
`clientPrivateKey`	Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados	""
`trustedCACertificate`	Valor codificado en Base64 del certificado de CA de confianza. Opcional. Usado para auth basada en certificados	""
`username`	Nombre de usuario para conectarse al clúster/SVM. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, consulta "Autentica Trident en un SVM backend usando credenciales de Active Directory".	
`password`	Contraseña para conectarte al clúster/SVM. Se usa para la autenticación basada en credenciales. Para la autenticación de Active Directory, consulta "Autentica Trident en un SVM backend usando credenciales de Active Directory".	
`storagePrefix`	Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede actualizar después de configurarlo Al usar ontap-nas-economy y un storagePrefix que tiene 24 caracteres o más, los qtrees no tendrán el prefijo de almacenamiento incorporado, aunque estará en el nombre del volumen.	"Trident"

Parámetro	Descripción	Predeterminado
`aggregate`	Agregado para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Para el `ontap-nas-flexgroup` driver, esta opción se ignora. Si no se asigna, cualquiera de los agregados disponibles se puede usar para aprovisionar un FlexGroup volumen. Cuando el agregado se actualiza en SVM, se actualiza automáticamente en Trident mediante el sondeo de SVM sin tener que reiniciar el Trident Controller. Cuando has configurado un agregado específico en Trident para aprovisionar volúmenes, si el agregado se renombra o se mueve fuera de la SVM, el backend pasará a estado fallido en Trident mientras sondea el agregado de la SVM. Debes cambiar el agregado por uno que esté presente en la SVM o eliminarlo por completo para que el backend vuelva a estar en línea. **No especifiques para sistemas de almacenamiento AFX.**	""
`limitAggregateUsage`	El aprovisionamiento falla si el uso supera este porcentaje. **No aplica a Amazon FSx for ONTAP**. **No especifiques para sistemas de almacenamiento AFX.**	"" (no aplicado por defecto)

Parámetro	Descripción	Predeterminado
flexgroupAggregateList	Lista de agregados para aprovisionamiento (opcional; si se configura, debe asignarse a la SVM). Todos los agregados asignados a la SVM se utilizan para aprovisionar un volumen FlexGroup. Compatible con el controlador de almacenamiento **ontap-nas-flexgroup**. Cuando la lista de agregados se actualiza en SVM, la lista se actualiza en Trident automáticamente mediante el sondeo de SVM sin tener que reiniciar el controlador Trident. Cuando has configurado una lista de agregados específica en Trident para aprovisionar volúmenes, si la lista de agregados se renombra o se mueve fuera de SVM, el backend pasará a estado de error en Trident mientras sondea el agregado de SVM. Debes cambiar la lista de agregados por una que esté presente en SVM o eliminarla por completo para que el backend vuelva a estar en línea.	""
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado es superior a este valor.	"" (no aplicado por defecto)
`debugTraceFlags`	Indicadores de depuración para usar al resolver problemas. Por ejemplo, {"api":false, "method":true} no uses `debugTraceFlags` a menos que estés resolviendo problemas y necesites un volcado de registro detallado.	null
`nasType`	Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. Si lo dejas en null, se usan volúmenes NFS por defecto. **Si lo especificas, siempre ponlo en `nfs` para sistemas de almacenamiento AFX**.	`nfs`
`nfsMountOptions`	Lista de opciones de montaje NFS separadas por comas. Las opciones de montaje para volúmenes persistentes de Kubernetes normalmente se especifican en las clases de almacenamiento, pero si no se especifican opciones de montaje en una clase de almacenamiento, Trident usará las opciones de montaje especificadas en el archivo de configuración del backend de almacenamiento. Si no se especifican opciones de montaje en la clase de almacenamiento ni en el archivo de configuración, Trident no establecerá ninguna opción de montaje en un volumen persistente asociado.	""

Parámetro	Descripción	Predeterminado
`qtreesPerFlexvol`	Máximo de Qtrees por FlexVol, debe estar en el rango [50, 300]	"200"
`smbShare`	Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o la CLI de ONTAP, un nombre para permitir que Trident cree el recurso compartido SMB, o puedes dejar el parámetro en blanco para evitar el acceso compartido común a los volúmenes. Este parámetro es opcional para ONTAP local. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP y no puede estar en blanco.	`smb-share`
`useREST`	Parámetro booleano para usar las ONTAP REST APIs. `useREST` Cuando se establece en `true`, Trident usa las ONTAP REST APIs para comunicarse con el backend; cuando se establece en `false`, Trident usa llamadas ONTAPI (ZAPI) para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontapi`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`. A partir de la versión Trident 24.06 y ONTAP 9.15.1 o posteriores, `useREST` se establece en `true` de forma predeterminada; cambia `useREST` a `false` para usar llamadas ONTAPI (ZAPI). **Si se especifica, siempre se establece en `true` para sistemas de almacenamiento AFX**.	`true` para ONTAP 9.15.1 o posterior, de lo contrario `false`.
`limitVolumePoolSize`	Tamaño máximo solicitable de FlexVol al utilizar Qtrees en el backend ontap-nas-economy.	"" (no aplicado por defecto)
`denyNewVolumePools`	Restringe `ontap-nas-economy` a los backends crear nuevos volúmenes FlexVol para contener sus Qtrees. Solo se usan FlexVols preexistentes para aprovisionar nuevos PVs.	
`adAdminUser`	Usuario o grupo de usuarios administradores de Active Directory con acceso total a los recursos compartidos SMB. Usa este parámetro para otorgar derechos de administrador al recurso compartido SMB con control total.	

Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

Parámetro	Descripción	Predeterminado
`spaceAllocation`	Asignación de espacio para Qtrees	"true"

Parámetro	Descripción	Predeterminado
`spaceReserve`	Modo de reserva de espacio; "ninguno" (fino) o "volumen" (grueso)	"none"
`snapshotPolicy`	Política de SnapVault a utilizar	"none"
`qosPolicy`	Grupo de políticas de QoS para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento/backend	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento/backend. No es compatible con ontap-nas-economy.	""
`snapshotReserve`	Porcentaje de volumen reservado para instantáneas	"0" si `snapshotPolicy` es "none", de lo contrario ""
`splitOnClone`	Divide un clon de su padre al momento de su creación	"false"
`encryption`	Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE".	"false"
`tieringPolicy`	Política de organización en niveles para usar "none"	
`unixPermissions`	Modo para nuevos volúmenes	"777" para volúmenes NFS; vacío (no aplicable) para volúmenes SMB
`snapshotDir`	Controla el acceso al `.snapshot` directorio	"verdadero" para NFSv4 "falso" para NFSv3
`exportPolicy`	Política de exportación a usar	"default"
`securityStyle`	Estilo de seguridad para nuevos volúmenes. NFS admite `mixed` y `unix` estilos de seguridad. SMB admite `mixed` y `ntfs` estilos de seguridad.	NFS predeterminado es `unix`. SMB predeterminado es `ntfs`.
`nameTemplate`	Plantilla para crear nombres de volúmenes personalizados.	""

Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite máximo para el rendimiento total de todas las cargas de trabajo.

Ejemplos de aprovisionamiento de volumen

Aquí tienes un ejemplo con valores predeterminados definidos:

```
---
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"
```

Para `ontap-nas` y `ontap-nas-flexgroups`, Trident ahora usa un nuevo cálculo para asegurarse de que el FlexVol tenga el tamaño correcto con el porcentaje de snapshotReserve y el PVC. Cuando el usuario solicita un PVC, Trident crea el FlexVol original con más espacio usando el nuevo cálculo. Este cálculo asegura que el usuario reciba el espacio escribible que pidió en el PVC, y no menos espacio del que solicitó. Antes de la v21.07, cuando el usuario solicitaba un PVC (por ejemplo, 5 GiB), con el snapshotReserve al 50 por ciento, solo obtenía 2.5 GiB de espacio escribible. Esto es porque lo que el usuario pedía era todo el volumen y `snapshotReserve` es un porcentaje de eso. Con Trident 21.07, lo que el usuario pide es el espacio escribible y Trident define el número de `snapshotReserve` como el porcentaje del volumen completo. Esto no aplica a `ontap-nas-economy`. Mira el siguiente ejemplo para ver cómo funciona esto:

El cálculo es el siguiente:

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

Para snapshotReserve = 50%, y una solicitud de PVC de 5 GiB, el tamaño total del volumen es 5/0.5 = 10 GiB y el tamaño disponible es 5 GiB, que es lo que el usuario pidió en la solicitud de PVC. El `volume show` comando debería mostrar resultados similares a este ejemplo:

Los backends existentes de instalaciones anteriores aprovisionarán volúmenes como se explicó arriba cuando actualices Trident. Para los volúmenes que creaste antes de actualizar, deberías redimensionar sus volúmenes para que se note el cambio. Por ejemplo, un PVC de 2 GiB con `snapshotReserve=50` antes generaba un volumen que proporcionaba 1 GiB de espacio escribible. Redimensionar el volumen a 3 GiB, por ejemplo, le da a la aplicación 3 GiB de espacio escribible en un volumen de 6 GiB.

Ejemplos de configuración mínima

Los siguientes ejemplos muestran configuraciones básicas que dejan la mayoría de los parámetros en sus valores predeterminados. Esta es la forma más fácil de definir un backend.

> Si estás usando Amazon FSx en NetApp ONTAP con Trident, la recomendación es especificar nombres DNS para los LIF en vez de direcciones IP.

Ejemplo de ONTAP NAS economy

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

Ejemplo de ONTAP NAS FlexGroup

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

Ejemplo de MetroCluster

Puedes configurar el backend para evitar tener que actualizar manualmente la definición del backend después de la conmutación de sitios y la conmutación de vuelta durante "Replicación y recuperación de SVM".

Para una conmutación de sitios sin interrupciones, especifica la SVM usando `managementLIF` y omite los parámetros `dataLIF` y `svm`. Por ejemplo:

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

Ejemplo de volúmenes SMB

```
---
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
```

Ejemplo de autenticación basada en certificados

Este es un ejemplo de configuración mínima de backend. `clientCertificate`, `clientPrivateKey` y `trustedCACertificate` (opcional, si usas una CA de confianza) se rellenan en `backend.json` y toman los valores codificados en base64 del certificado de cliente, la clave privada y el certificado de CA de confianza, respectivamente.

```
---
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_
```

Ejemplo de política de exportación automática

Este ejemplo muestra cómo puedes indicarle a Trident que use políticas de exportación dinámicas para crear y gestionar la política de exportación automáticamente. Esto funciona igual para los controladores `ontap-nas-economy` y `ontap-nas-flexgroup`.

```
---
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
```

Ejemplo de direcciones IPv6

Este ejemplo muestra `managementLIF` usando una dirección IPv6.

```
---
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
```

Ejemplo de Amazon FSx for ONTAP usando volúmenes SMB

El `smbShare` parámetro es necesario para FSx for ONTAP usando volúmenes SMB.

```
---
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_
```

Ejemplo de configuración de backend con 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}}"
```

Ejemplos de backends con pools virtuales

En los archivos de definición de backend de ejemplo que se muestran a continuación, se establecen valores predeterminados específicos para todos los pools de almacenamiento, como `spaceReserve` en ninguno, `spaceAllocation` en falso y `encryption` en falso. Los pools virtuales se definen en la sección de almacenamiento.

Trident establece etiquetas de aprovisionamiento en el campo "Comentarios". Los comentarios se establecen en FlexVol para `ontap-nas` o en FlexGroup para `ontap-nas-flexgroup`. Trident copia todas las etiquetas presentes en un pool virtual al volumen de almacenamiento al aprovisionar. Para mayor comodidad, los administradores de almacenamiento pueden definir etiquetas por pool virtual y agrupar volúmenes por etiqueta.

En estos ejemplos, algunos de los pools de almacenamiento establecen sus propios `spaceReserve`, `spaceAllocation` y `encryption` valores, y algunos pools anulan los valores predeterminados.

Ejemplo de ONTAP NAS

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

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

Ejemplo de ONTAP NAS FlexGroup

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

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

Ejemplo de ONTAP NAS economy

```
---
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"
```

Asigna backends a StorageClasses

Las siguientes definiciones de StorageClass hacen referencia a Ejemplos de backends con pools virtuales. Usando el campo `parameters.selector`, cada StorageClass indica qué pools virtuales pueden usarse para alojar un volumen. El volumen tendrá los aspectos definidos en el pool virtual elegido.

- El `protection-gold` StorageClass se asignará al primer y segundo pool virtual en el `ontap-nas-flexgroup` backend. Estos son los únicos pools que ofrecen protección de nivel oro.

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

- El `protection-not-gold` StorageClass se asignará al tercer y cuarto pool virtual en el `ontap-nas-flexgroup` backend. Estos son los únicos pools que ofrecen un nivel de protección distinto al gold.

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

- El `app-mysqldb` StorageClass se mapeará al cuarto pool virtual en el `ontap-nas` backend. Este es el único pool que ofrece configuración de pool de almacenamiento para app tipo mysqldb.

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

- El `protection-silver-creditpoints-20k` StorageClass se asignará a la tercera reserva virtual en el `ontap-nas-flexgroup` backend. Esta es la única reserva que ofrece protección de nivel plata y 20000 puntos de crédito.

```
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"
```

- El `creditpoints-5k` StorageClass se asignará al tercer pool virtual en el `ontap-nas` backend y al segundo pool virtual en el `ontap-nas-economy` backend. Estas son las únicas ofertas de pool con 5000 creditpoints.

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

Trident decidirá qué grupo virtual se selecciona y se asegurará de que se cumpla el requisito de almacenamiento.

Actualiza `dataLIF` después de la configuración inicial

Puedes cambiar el dataLIF después de la configuración inicial ejecutando el siguiente comando para proporcionar el nuevo archivo JSON del backend con el dataLIF actualizado.

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

Si los PVC están conectados a uno o varios pods, tienes que desactivar todos los pods correspondientes y luego volverlos a activar para que el nuevo dataLIF surta efecto.

Ejemplos seguros de SMB

Configuración del backend con el controlador ontap-nas

```
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
```

Configuración del backend con el controlador ontap-nas-economy

```
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
```

Configuración de backend con storage pool

```
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
```

Ejemplo de clase de almacenamiento con el controlador ontap-nas

```
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
```

Asegúrate de agregar `annotations` para habilitar SMB seguro. SMB seguro no funciona sin las anotaciones, independientemente de las configuraciones establecidas en el Backend o PVC.

Ejemplo de clase de almacenamiento con el controlador ontap-nas-economy

```
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
```

Ejemplo de PVC con un único usuario AD

```
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
```

Ejemplo de PVC con varios usuarios AD

```
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

Usa Trident con Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" es un servicio de AWS completamente administrado que permite a los clientes lanzar y ejecutar sistemas de archivos con el sistema operativo de almacenamiento NetApp ONTAP. FSx for ONTAP te permite aprovechar las características, el rendimiento y las capacidades administrativas de NetApp que ya conoces, mientras aprovechas la simplicidad, agilidad, seguridad y escalabilidad de almacenar datos en AWS. FSx for ONTAP es compatible con las características del sistema de archivos ONTAP y las API de administración.

Puedes integrar tu sistema de archivos Amazon FSx for NetApp ONTAP con Trident para asegurarte de que los clústeres de Kubernetes que se ejecutan en Amazon Elastic Kubernetes Service (EKS) puedan aprovisionar volúmenes persistentes de bloques y archivos respaldados por ONTAP.

Un sistema de archivos es el recurso principal en Amazon FSx, similar a un clúster de ONTAP en las instalaciones. Dentro de cada SVM puedes crear uno o varios volúmenes, que son contenedores de datos que

almacenan los archivos y carpetas en tu sistema de archivos. Con Amazon FSx for NetApp ONTAP se proporcionará como un sistema de archivos gestionado en la nube. El nuevo tipo de sistema de archivos se llama **NetApp ONTAP**.

Al usar Trident con Amazon FSx for NetApp ONTAP, puedes asegurarte de que los clústeres de Kubernetes que se ejecutan en Amazon Elastic Kubernetes Service (EKS) puedan aprovisionar volúmenes persistentes de bloques y archivos respaldados por ONTAP.

Requisitos

Además de "Requisitos de Trident", para integrar FSx for ONTAP con Trident, necesitas:

- Un clúster de Amazon EKS existente o un clúster de Kubernetes autoadministrado con `kubectl` instalado.
- Un sistema de archivos Amazon FSx para NetApp ONTAP existente y una máquina virtual de almacenamiento (SVM) a la que se puede acceder desde los nodos de trabajo de tu clúster.
- Nodos worker que están preparados para "NFS o iSCSI".

> Asegúrate de seguir los pasos de preparación de nodo necesarios para Amazon Linux y Ubuntu "Amazon Machine Images" (AMIs) según tu tipo de AMI de EKS.

Consideraciones

- Volúmenes SMB:
  - Los volúmenes SMB son compatibles usando solo el `ontap-nas` driver.
  - Los volúmenes SMB no son compatibles con el complemento Trident EKS.
  - Trident solo admite volúmenes SMB montados en pods que se ejecutan en nodos Windows. Consulta "Prepárate para aprovisionar volúmenes SMB" para más detalles.
- Antes de Trident 24.02, los volúmenes creados en sistemas de archivos de Amazon FSx que tenían copias de seguridad automáticas habilitadas no podían ser eliminados por Trident. Para evitar este problema en Trident 24.02 o versiones posteriores, especifica el `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` y AWS `secretKey` en el archivo de configuración del backend para AWS FSx for ONTAP.

> Si estás especificando una función de IAM para Trident, entonces puedes omitir especificar los campos `apiRegion`, `apiKey` y `secretKey` explícitamente para Trident. Para más información, consulta "Opciones de configuración de FSx for ONTAP y ejemplos".

Uso simultáneo de Trident SAN/iSCSI y el controlador EBS-CSI

Si planeas usar controladores ontap-san (por ejemplo, iSCSI) con AWS (EKS, ROSA, EC2 o cualquier otra instancia), la configuración de multivía requerida en los nodos podría entrar en conflicto con el controlador CSI de Amazon Elastic Block Store (EBS). Para asegurarte de que la multivía funcione sin interferir con los discos EBS en el mismo nodo, necesitas excluir EBS en tu configuración de multivía. Este ejemplo muestra un `multipath.conf` archivo que incluye la configuración requerida de Trident y excluye los discos EBS de la multivía:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Autenticación

Trident ofrece dos modos de autenticación.

- Basado en credenciales (recomendado): almacena las credenciales de forma segura en AWS Secrets Manager. Puedes usar el `fsxadmin` usuario para tu sistema de archivos o el `vsadmin` usuario configurado para tu SVM.

> Trident espera ejecutarse como un `vsadmin` usuario de SVM o como un usuario con un nombre diferente que tenga el mismo rol. Amazon FSx for NetApp ONTAP tiene un `fsxadmin` usuario que es un reemplazo limitado del usuario de clúster de ONTAP `admin`. Te recomendamos mucho usar `vsadmin` con Trident.

- Basado en certificado: Trident se comunicará con la SVM en tu sistema de archivos FSx usando un certificado instalado en tu SVM.

Para obtener detalles sobre cómo habilitar la autenticación, consulta la autenticación para tu tipo de controlador:

- "Autenticación NAS de ONTAP"
- "Autenticación SAN de ONTAP"

Imágenes de máquina de Amazon (AMIs) probadas

El clúster EKS es compatible con varios sistemas operativos, pero AWS ha optimizado ciertas Amazon Machine Images (AMIs) para contenedores y EKS. Las siguientes AMIs se han probado con NetApp Trident 25.02.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economy
AL2023_x86_64_STANDARD	Sí	Sí	Sí	Sí
AL2_x86_64	Sí	Sí	Sí*	Sí*
BOTTLEROCKET_x86_64	Sí**	Sí	N/A	N/A
AL2023_ARM_64_STANDARD	Sí	Sí	Sí	Sí
AL2_ARM_64	Sí	Sí	Sí*	Sí*

BOTTLEROCKET_ARM_64	Sí**	Sí	N/A	N/A

- * No se puede eliminar el PV sin reiniciar el nodo
- ** No funciona con NFSv3 con Trident versión 25.02.

> Si la AMI que buscas no aparece aquí, no significa que no sea compatible; simplemente significa que no se ha probado. Esta lista sirve como guía para las AMI que se sabe que funcionan.

Pruebas realizadas con:

- Versión de EKS: 1.32
- Método de instalación: Helm 25.06 y como AWS add-On 25.06
- Para NAS se probaron tanto NFSv3 como NFSv4.1.
- Para SAN solo se probó iSCSI, no NVMe-oF.

Pruebas realizadas:

- Crear: clase de almacenamiento, pvc, pod
- Eliminar: pod, pvc (regular, qtree/lun – economy, NAS con backup de AWS)

Encuentra más información

- "Documentación de Amazon FSx para NetApp ONTAP"
- "Entrada de blog sobre Amazon FSx para NetApp ONTAP"

Crea un rol de IAM y un secreto de AWS

Puedes configurar pods de Kubernetes para acceder a recursos de AWS autenticándote como un rol de AWS IAM en lugar de proporcionar credenciales de AWS explícitas.

> Para autenticarse mediante una función de AWS IAM, debes tener un clúster de Kubernetes implementado usando EKS.

Crear secreto de AWS Secrets Manager

Como Trident va a emitir APIs contra un vserver de FSx para gestionar el almacenamiento por ti, necesitará credenciales para hacerlo. La forma segura de pasar esas credenciales es a través de un secreto de AWS Secrets Manager. Así que, si aún no tienes uno, tendrás que crear un secreto de AWS Secrets Manager que contenga las credenciales para la cuenta vsadmin.

Este ejemplo crea un secreto de AWS Secrets Manager para almacenar las credenciales de Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Crear política de IAM

Trident también necesita permisos de AWS para funcionar correctamente. Por lo tanto, necesitas crear una política que le dé a Trident los permisos que necesita.

Los siguientes ejemplos crean una política de IAM mediante el 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"
```

Ejemplo de política JSON:

```
{
  "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"
}
```

Crea una identidad de pod o un rol de IAM para la asociación de cuentas de servicio (IRSA)

Puedes configurar una cuenta de servicio de Kubernetes para que asuma un rol de AWS Identity and Access Management (IAM) con EKS Pod Identity o IAM role for Service account association (IRSA). Cualquier pod que esté configurado para usar la cuenta de servicio podrá acceder a cualquier servicio de AWS al que el rol

tenga permisos de acceso.

Identidad de pod

Las asociaciones de Amazon EKS Pod Identity proporcionan la capacidad de administrar credenciales para tus aplicaciones, de forma similar a como los perfiles de instancias de Amazon EC2 proporcionan credenciales a las instancias de Amazon EC2.

Instala Pod Identity en tu clúster EKS:

Puedes crear una identidad de Pod a través de la consola de AWS o usando el siguiente comando de la AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Para más información consulta "Configura el agente de identidad de Amazon EKS Pod".

Crea trust-relationship.json:

Crea trust-relationship.json para permitir que EKS Service Principal asuma este rol para Pod Identity. Luego, crea un rol con esta política de confianza:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

archivo trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Adjunta la política de rol al rol IAM:

Adjunta la política de rol del paso anterior al rol IAM que creaste:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Crea una asociación de identidad de pod:

Crea una asociación de identidad de pod entre el rol IAM y la cuenta de servicio de Trident (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
```

Función IAM para la asociación de cuenta de servicio (IRSA)

Usando la AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

Archivo trust-relationship.json:

```
{
  "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"
        }
      }
    }
  ]
}
```

Actualiza los siguientes valores en el archivo `trust-relationship.json`:

- **<account_id>** - Tu ID de cuenta AWS
- **<oidc_provider>** - El OIDC de tu cluster EKS. Puedes obtener el oidc_provider ejecutando:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Adjunta el rol IAM con la política IAM:

Una vez que el rol ha sido creado, adjunta la política (que fue creada en el paso anterior) al rol usando este comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verifica que el proveedor OICD está asociado:

Verifica que tu proveedor OIDC está asociado con tu cluster. Puedes verificarlo usando este comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si la salida está vacía, usa el siguiente comando para asociar IAM OIDC a tu clúster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Si usas eksctl, usa el siguiente ejemplo para crear una función IAM para la cuenta de servicio en EKS:

```
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
```

Instala Trident

Trident agiliza la administración del almacenamiento de Amazon FSx para NetApp ONTAP en Kubernetes para que tus desarrolladores y administradores se enfoquen en la puesta en marcha de aplicaciones.

Puedes instalar Trident usando uno de los siguientes métodos:

- Helm
- Complemento EKS

Si quieres hacer uso de la funcionalidad de instantáneas, instala el complemento controlador de instantáneas CSI. Consulta "Habilita la funcionalidad de instantáneas para volúmenes CSI" para más información.

Instala Trident con helm

Identidad de pod

1. Agrega el repositorio Trident Helm:

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

2. Instala Trident usando el siguiente ejemplo:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Puedes usar el comando `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación y número de revisión.

```
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
```

Asociación de cuenta de servicio (IRSA)

1. Agrega el repositorio Trident Helm:

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

2. Establece los valores de **proveedor de nube** e **identidad de nube**:

```
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
```

Puedes usar el comando `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación y número de revisión.

```
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
```

Si estás planeando usar iSCSI, asegúrate de que iSCSI esté habilitado en tu máquina cliente. Si estás usando AL2023 Worker node OS, puedes automatizar la instalación del cliente iSCSI añadiendo el parámetro node prep en la instalación de helm:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

Instala Trident mediante el add-on EKS

El complemento Trident EKS incluye los últimos parches de seguridad, correcciones de errores y está validado por AWS para funcionar con Amazon EKS. El complemento EKS te permite asegurarte de forma constante de que tus clústeres de Amazon EKS sean seguros y estables, y reducir la cantidad de trabajo que necesitas hacer para instalar, configurar y actualizar complementos.

Prerrequisitos

Asegúrate de tener lo siguiente antes de configurar el complemento Trident para AWS EKS:

- Una cuenta de clúster de Amazon EKS con suscripción de complemento
- Permisos de AWS para AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) o Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nodo: AMD o ARM
- Un sistema de archivos existente de Amazon FSx para NetApp ONTAP

Habilita el complemento Trident para AWS

Consola de gestión

1. Abre la consola de Amazon EKS en https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster para el que quieres configurar el complemento NetApp Trident CSI.
4. Selecciona **Add-ons** y luego **Get more add-ons**.
5. Sigue estos pasos para seleccionar el complemento:
   a. Desplázate hasta la sección **AWS Marketplace add-ons** y escribe **"Trident"** en el cuadro de búsqueda.
   b. Selecciona la casilla de comprobación en la esquina superior derecha del cuadro Trident by NetApp.
   c. Selecciona **Siguiente**.
6. En la página de configuración **Configurar add-ons seleccionados**, haz lo siguiente:

   Omite estos pasos si estás usando la asociación Pod Identity.

   a. Selecciona la **Version** que te gustaría usar.
   b. Si estás usando la autenticación IRSA, asegúrate de establecer los valores de configuración disponibles en la configuración opcional:
      - Selecciona la **Version** que te gustaría usar.
      - Sigue el **Add-on configuration schema** y establece el parámetro **configurationValues** en la sección **Configuration values** al role-arn que creaste en el paso anterior (el valor debe tener el siguiente formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Si seleccionas Override como método de resolución de conflictos, una o más de las configuraciones del add-on existente pueden sobrescribirse con las configuraciones del Amazon EKS add-on. Si no habilitas esta opción y hay un conflicto con tus configuraciones existentes, la operación falla. Puedes usar el mensaje de error resultante para solucionar el conflicto. Antes de seleccionar esta opción, asegúrate de que el Amazon EKS add-on no gestione configuraciones que necesites administrar tú mismo.

7. Elige **Siguiente**.
8. En la página **Revisar y agregar**, elige **Crear**.

   Después de que se complete la instalación del complemento, verás tu complemento instalado.

AWS CLI

1. Crea el `add-on.json` file:

Para la identidad del pod, usa el siguiente formato:

Usa la

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Para la autenticación de IRSA, usa el siguiente formato:

```
{
  "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"
  }
}
```

Reemplaza `<role ARN>` con el ARN del rol que se creó en el paso anterior.

2. Instala el complemento Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

El siguiente comando de ejemplo instala el complemento Trident EKS:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Actualiza el add-on Trident EKS

Consola de gestión

1. Abre la consola de Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster para el que quieres actualizar el software complementario Trident CSI de NetApp.
4. Selecciona la pestaña **Add-ons**.
5. Selecciona **Trident by NetApp** y luego selecciona **Editar**.
6. En la página **Configura Trident por NetApp**, haz lo siguiente:
   a. Selecciona la **Version** que te gustaría usar.
   b. Expande las **Configuraciones de configuración opcionales** y modifícalas según sea necesario.
   c. Selecciona **Guardar cambios**.

AWS CLI

El siguiente ejemplo actualiza el add-on EKS:

```
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

- Comprueba la versión actual de tu FSxN Trident CSI add-on. Reemplaza `my-cluster` con el nombre de tu clúster.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  Ejemplo de salida:

```
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'"}
```

- Actualiza el software complementario a la versión que aparece bajo UPDATE AVAILABLE en la salida del paso anterior.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

Si quitas la opción `--force` y alguna de las configuraciones del complemento de Amazon EKS entra en conflicto con tus configuraciones actuales, entonces la actualización del complemento de Amazon EKS falla; recibes un mensaje de error para ayudarte a resolver el conflicto. Antes de especificar esta opción, asegúrate de que el complemento de Amazon EKS no administre configuraciones que necesites administrar, porque esas configuraciones se sobrescriben con esta opción. Para más información sobre otras opciones para esta configuración, consulta "Complementos". Para más información sobre la administración de campos de Kubernetes de Amazon EKS, consulta "Gestión de campos de Kubernetes".

Desinstala/elimina el complemento Trident EKS

Tienes dos opciones para eliminar un complemento de Amazon EKS:

- **Conservar el software complementario en tu clúster** – Esta opción elimina la administración de cualquier configuración por parte de Amazon EKS. También elimina la capacidad de Amazon EKS para notificarte sobre actualizaciones y actualizar automáticamente el Amazon EKS add-on después de que inicies una actualización. Sin embargo, conserva el software complementario en tu clúster. Esta opción hace que el add-on sea una instalación autogestionada, en lugar de un Amazon EKS add-on. Con esta opción, no hay tiempo de inactividad para el add-on. Retén la `--preserve` opción en el comando para conservar el add-on.
- **Elimina el software complementario por completo de tu clúster** – NetApp recomienda que elimines el complemento de Amazon EKS de tu clúster solo si no hay recursos en tu clúster que dependan de él. Elimina la opción `--preserve` del comando `delete` para eliminar el complemento.

Si el software complementario tiene una cuenta IAM asociada, la cuenta IAM no se elimina.

Consola de gestión

1. Abre la consola de Amazon EKS en https://console.aws.amazon.com/eks/home#/clusters.
2. En el panel de navegación izquierdo, selecciona **Clusters**.
3. Selecciona el nombre del clúster del que quieres quitar el NetApp Trident CSI add-on.
4. Selecciona la pestaña **Complementos** y luego selecciona **Trident by NetApp**.*
5. Selecciona **Eliminar**.
6. En el cuadro de diálogo **Remove netapp_trident-operator confirmation**, haz lo siguiente:
   a. Si quieres que Amazon EKS deje de administrar la configuración del software complementario, selecciona **Conservar en el clúster**. Haz esto si quieres conservar el software complementario en tu clúster para que puedas administrar tú mismo todas las configuraciones del complemento.
   b. Ingresa **netapp_trident-operator**.
   c. Selecciona **Eliminar**.

AWS CLI

Reemplaza `my-cluster` con el nombre de tu clúster y luego ejecuta el siguiente comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

El siguiente comando desinstala el complemento Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configura el backend de almacenamiento

Integración de controladores SAN y NAS de ONTAP

Para crear un backend de almacenamiento, necesitas crear un archivo de configuración en formato JSON o YAML. El archivo debe especificar el tipo de almacenamiento que quieres (NAS o SAN), el sistema de archivos, la SVM de la que se obtendrá y cómo autenticarse con ella. El siguiente ejemplo muestra cómo definir almacenamiento basado en NAS y usar un secreto de AWS para guardar las credenciales de la SVM que quieres usar:

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"
    }
  }
}
```

Ejecuta los siguientes comandos para crear y validar la configuración de backend de Trident (TBC):

- Crea la configuración del backend de Trident (TBC) desde el archivo yaml y ejecuta el siguiente comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Valida que la configuración de backend de Trident (TBC) se creó correctamente:

```
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
```

Detalles del controlador de FSx for ONTAP

Puedes integrar Trident con Amazon FSx for NetApp ONTAP usando los siguientes controladores:

- `ontap-san`: Cada PV aprovisionado es un LUN dentro de su propio volumen de Amazon FSx for NetApp ONTAP. Recomendado para almacenamiento en bloques.
- `ontap-nas`: Cada PV aprovisionado es un volumen completo de Amazon FSx for NetApp ONTAP. Recomendado para NFS y SMB.
- `ontap-san-economy`: cada PV aprovisionado es un LUN con una cantidad configurable de LUN por volumen de Amazon FSx for NetApp ONTAP.
- `ontap-nas-economy`: Cada PV aprovisionado es un qtree, con una cantidad configurable de qtrees por volumen de Amazon FSx para NetApp ONTAP.
- `ontap-nas-flexgroup`: cada PV aprovisionado es un volumen completo de Amazon FSx for NetApp ONTAP FlexGroup.

Para obtener detalles del controlador, consulta "Controladores NAS" y "Controladores SAN".

Una vez creado el archivo de configuración, ejecuta este comando para crearlo dentro de tu EKS:

```
kubectl create -f configuration_file
```

Para verificar el estado, ejecuta este comando:

```
kubectl get tbc -n trident
```

```
NAME                     BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

Configuración avanzada de backend y ejemplos

Consulta la siguiente tabla para ver las opciones de configuración del backend:

Parámetro	Descripción	Ejemplo
`version`		Siempre 1
`storageDriverName`	Nombre del controlador de almacenamiento	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nombre personalizado o el backend de almacenamiento	Nombre del driver + "_" + dataLIF
`managementLIF`	Dirección IP de un clúster o de una LIF de administración de SVM. Se puede especificar un nombre de dominio completo (FQDN). Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el indicador IPv6. Las direcciones IPv6 deben definirse entre corchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si proporcionas el `fsxFilesystemID` bajo el campo `aws`, no necesitas proporcionar el `managementLIF` porque Trident obtiene la información de SVM `managementLIF` desde AWS. Así que debes proporcionar credenciales para un usuario bajo la SVM (por ejemplo: vsadmin) y el usuario debe tener el rol `vsadmin`.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parámetro	Descripción	Ejemplo
`dataLIF`	Dirección IP del protocolo LIF. **ONTAP NAS drivers**: NetApp recomienda especificar dataLIF. Si no se proporciona, Trident obtiene los dataLIF de la SVM. Puedes especificar un nombre de dominio completo (FQDN) para las operaciones de montaje NFS, lo que te permite crear un DNS round-robin para equilibrar la carga entre varios dataLIF. Se puede cambiar después de la configuración inicial. Consulta . **ONTAP SAN drivers**: no especificar para iSCSI. Trident usa ONTAP Selective LUN Map para descubrir los iSCSI LIF necesarios para establecer una sesión multipath. Se genera una advertencia si dataLIF se define explícitamente. Se puede configurar para usar direcciones IPv6 si Trident se instaló usando el flag IPv6. Las direcciones IPv6 deben definirse entre corchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Habilita la creación y actualización automática de políticas de exportación [Boolean]. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente.	`false`
`autoExportCIDRs`	Lista de CIDR para filtrar las direcciones IP de los nodos de Kubernetes cuando `autoExportPolicy` está habilitado. Usando las opciones `autoExportPolicy` y `autoExportCIDRs`, Trident puede gestionar las políticas de exportación automáticamente.	"["0.0.0.0/0", "::/0"]"
`labels`	Conjunto de etiquetas arbitrarias con formato JSON para aplicar en volúmenes	""
`clientCertificate`	Valor codificado en Base64 del certificado del cliente. Usado para autenticación basada en certificados	""

Parámetro	Descripción	Ejemplo
`clientPrivateKey`	Valor codificado en Base64 de la clave privada del cliente. Usado para auth basada en certificados	""
`trustedCACertificate`	Valor codificado en Base64 del certificado de CA de confianza. Opcional. Se usa para la autenticación basada en certificados.	""
`username`	Nombre de usuario para conectarte al clúster o SVM. Se usa para la autenticación basada en credenciales. Por ejemplo, vsadmin.	
`password`	Contraseña para conectarse al clúster o SVM. Se utiliza para la autenticación basada en credenciales.	
`svm`	Máquina virtual de almacenamiento que vas a usar	Derivado si se especifica un LIF de gestión de SVM.
`storagePrefix`	Prefijo utilizado al aprovisionar nuevos volúmenes en la SVM. No se puede modificar después de la creación. Para actualizar este parámetro, tendrás que crear un nuevo backend.	`trident`
`limitAggregateUsage`	**No especifiques para Amazon FSx for NetApp ONTAP.** Los `fsxadmin` y `vsadmin` no tienen los permisos necesarios para recuperar el uso agregado y limitarlo usando Trident.	No uses.
`limitVolumeSize`	Falla el aprovisionamiento si el tamaño del volumen solicitado supera este valor. También restringe el tamaño máximo de los volúmenes que administra para qtrees y LUNs, y la `qtreesPerFlexvol` opción permite personalizar el número máximo de qtrees por FlexVol volume	"" (no aplicado por defecto)
`lunsPerFlexvol`	LUNs máximos por FlexVol volume, deben estar en el rango [50, 200]. Solo SAN.	“100”

Parámetro	Descripción	Ejemplo
`debugTraceFlags`	Indicadores de depuración para usar al resolver problemas. Por ejemplo, {"api":false, "method":true} no uses `debugTraceFlags` a menos que estés resolviendo problemas y necesites un volcado de registro detallado.	null
`nfsMountOptions`	Lista de opciones de montaje NFS separadas por comas. Las opciones de montaje para volúmenes persistentes de Kubernetes normalmente se especifican en las clases de almacenamiento, pero si no se especifican opciones de montaje en una clase de almacenamiento, Trident usará las opciones de montaje especificadas en el archivo de configuración del backend de almacenamiento. Si no se especifican opciones de montaje en la clase de almacenamiento ni en el archivo de configuración, Trident no establecerá ninguna opción de montaje en un volumen persistente asociado.	""
`nasType`	Configura la creación de volúmenes NFS o SMB. Las opciones son `nfs`, `smb` o null. **Debes establecerlo en `smb` para volúmenes SMB.** Si lo configuras en null, se usarán volúmenes NFS por defecto.	`nfs`
`qtreesPerFlexvol`	Máximo de qtrees por FlexVol volume, debe estar en el rango [50, 300]	`"200"`
`smbShare`	Puedes especificar uno de los siguientes: el nombre de un recurso compartido SMB creado usando Microsoft Management Console o ONTAP CLI, o un nombre para permitir que Trident cree el recurso compartido SMB. Este parámetro es obligatorio para los backends de Amazon FSx for ONTAP.	`smb-share`

Parámetro	Descripción	Ejemplo
`useREST`	Parámetro booleano para usar las ONTAP REST APIs. Cuando se establece en `true`, Trident usará las ONTAP REST APIs para comunicarse con el backend. Esta función requiere ONTAP 9.11.1 y versiones posteriores. Además, el rol de inicio de sesión de ONTAP utilizado debe tener acceso a la aplicación `ontap`. Esto se cumple con los roles predefinidos `vsadmin` y `cluster-admin`.	`false`
`aws`	Puedes especificar lo siguiente en el archivo de configuración para AWS FSx for ONTAP: - `fsxFilesystemID`: especifica el ID del sistema de archivos de AWS FSx. - `apiRegion`: nombre de la región de la API de AWS. - `apikey`: clave de API de AWS. - `secretKey`: clave secreta de AWS.	`""` `""` `""`
`credentials`	Especifica las credenciales de FSx SVM que se van a guardar en AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) del secreto, que contiene las credenciales de SVM. - `type`: Establécelo en `awsarn`. Consulta "Crea un secreto de AWS Secrets Manager" para más información.	

Opciones de configuración de backend para aprovisionar volúmenes

Puedes controlar el aprovisionamiento predeterminado usando estas opciones en la `defaults` sección de la configuración. Por ejemplo, mira los ejemplos de configuración abajo.

Parámetro	Descripción	Predeterminado
`spaceAllocation`	Asignación de espacio para LUNs	`true`
`spaceReserve`	Modo de reserva de espacio; "ninguno" (fino) o "volumen" (grueso)	`none`
`snapshotPolicy`	Política de SnapVault a utilizar	`none`

Parámetro	Descripción	Predeterminado
`qosPolicy`	Grupo de políticas de QoS para asignar a los volúmenes creados. Elige uno de qosPolicy o adaptiveQosPolicy por pool de almacenamiento o backend. Usar grupos de políticas de QoS con Trident requiere ONTAP 9.8 o una versión posterior. Deberías usar un grupo de políticas de QoS no compartido y asegurarte de que el grupo de políticas se aplique a cada componente individualmente. Un grupo de políticas de QoS compartido impone el límite máximo para el rendimiento total de todas las cargas de trabajo.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptativo para asignar a los volúmenes creados. Elige una de qosPolicy o adaptiveQosPolicy por pool de almacenamiento o backend. No compatible con ontap-nas-economy.	""
`snapshotReserve`	Porcentaje de volumen reservado para snapshots "0"	Si `snapshotPolicy` es `none`, `else` "
`splitOnClone`	Divide un clon de su padre al momento de su creación	`false`
`encryption`	Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE".	`false`
`luksEncryption`	Habilita el cifrado LUKS. Consulta "Usa Linux Unified Key Setup (LUKS)". Solo SAN.	""
`tieringPolicy`	Política de tiering a usar `none`	
`unixPermissions`	Modo para nuevos volúmenes. **Dejar vacío para volúmenes SMB.**	""

Parámetro	Descripción	Predeterminado
`securityStyle`	Estilo de seguridad para nuevos volúmenes. NFS admite `mixed` y `unix` estilos de seguridad. SMB admite `mixed` y `ntfs` estilos de seguridad.	NFS predeterminado es `unix`. SMB predeterminado es `ntfs`.

Aprovisionar volúmenes SMB

Puedes aprovisionar volúmenes SMB usando el `ontap-nas` driver. Antes de completar Integración de controladores SAN y NAS de ONTAP completa estos pasos: "Prepárate para aprovisionar volúmenes SMB".

Configura una clase de almacenamiento y PVC

Configura un objeto StorageClass de Kubernetes y crea la clase de almacenamiento para indicarle a Trident cómo aprovisionar volúmenes. Crea un PersistentVolumeClaim (PVC) que use el StorageClass de Kubernetes configurado para solicitar acceso al PV. Luego puedes montar el PV en un pod.

Crear una clase de almacenamiento

Configura un objeto StorageClass de Kubernetes

El "Objeto de Kubernetes StorageClass" objeto identifica a Trident como el aprovisionador que se usa para esa clase y le indica a Trident cómo aprovisionar un volumen. Usa este ejemplo para configurar Storageclass para volúmenes que usan NFS (consulta la sección de atributos de Trident a continuación para la lista completa de atributos):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Usa este ejemplo para configurar Storageclass para volúmenes que usan iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Para aprovisionar volúmenes NFSv3 en AWS Bottlerocket, agrega lo requerido `mountOptions` a la clase de almacenamiento:

```
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
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

Crear una clase de almacenamiento

Pasos

1. Este es un objeto de Kubernetes, así que usa `kubectl` para crearlo en Kubernetes.

   ```
   kubectl create -f storage-class-ontapnas.yaml
   ```

2. Ahora deberías ver una clase de almacenamiento **basic-csi** tanto en Kubernetes como en Trident, y Trident debería haber descubierto los pools en el backend.

   ```
   kubectl get sc basic-csi
   ```

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

Crea el PVC

Una "*PersistentVolumeClaim*" (PVC) es una solicitud de acceso al PersistentVolume en el clúster.

El PVC se puede configurar para solicitar almacenamiento de un tamaño o modo de acceso determinados. Usando el StorageClass asociado, el administrador del clúster puede controlar más que solo el tamaño y el modo de acceso de PersistentVolume, como el rendimiento o el nivel de servicio.

Después de crear el PVC, puedes montar el volumen en un pod.

Manifiestos de muestra

Manifiestos de muestra de PersistentVolumeClaim

Estos ejemplos muestran opciones básicas de configuración de PVC.

PVC con acceso RWX

Este ejemplo muestra un PVC básico con acceso RWX que está asociado con un StorageClass llamado `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

Ejemplo de PVC usando iSCSI

Este ejemplo muestra un PVC básico para iSCSI con acceso RWO que está asociado con un StorageClass llamado `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 1Gi
storageClassName: protection-gold
```

Crear PVC

Pasos

1. Crea el PVC.

```
kubectl create -f pvc.yaml
```

2. Verifica el estado del PVC.

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                           5m
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

Atributos de Trident

Estos parámetros determinan qué grupos de almacenamiento administrados por Trident se deben utilizar para aprovisionar volúmenes de un tipo dado.

Atributo	Tipo	Valores	Oferta	Solicitud	Con el apoyo de
medios[1]	cadena	hdd, híbrido, ssd	El pool contiene medios de este tipo; híbrido significa ambos	Tipo de medio especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	cadena	delgado, grueso	Pool admite este método de aprovisionamiento	Método de aprovisionamiento especificado	grueso: todo ontap; fino: todo ontap & solidfire-san
backendType	cadena	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertenece a este tipo de backend	Backend especificado	Todos los drivers
instantáneas	bool	verdadero, falso	El pool admite volúmenes con instantáneas	Volumen con instantáneas habilitadas	ontap-nas, ontap-san, solidfire-san
clones	bool	verdadero, falso	El pool admite la clonación de volúmenes	Volumen con clones habilitados	ontap-nas, ontap-san, solidfire-san

Atributo	Tipo	Valores	Oferta	Solicitud	Con el apoyo de
cifrado	bool	verdadero, falso	El pool admite volúmenes cifrados	Volumen con cifrado habilitado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	entero	entero positivo	El pool es capaz de garantizar IOPS en este rango	Volumen garantizó estas IOPS	solidfire-san

[1]: No compatible con los sistemas ONTAP Select

Poner en marcha una aplicación de muestra

Una vez creada la clase de almacenamiento y el PVC, puedes montar el PV en un pod. Esta sección muestra el comando de ejemplo y la configuración para conectar el PV a un pod.

Pasos

1. Monta el volumen en un pod.

```
kubectl create -f pv-pod.yaml
```

Estos ejemplos muestran configuraciones básicas para conectar el PVC a un pod: **configuración básica**:

```
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
```

Puedes monitorear el progreso usando `kubectl get pod --watch`.

2. Verifica que el volumen esté montado en `/my/mount/path`.

```
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
```

Ahora puedes eliminar el Pod. La aplicación del Pod ya no existirá, pero el volumen permanecerá.

```
kubectl delete pod pv-pod
```

Configura el complemento Trident EKS en un clúster EKS

NetApp Trident optimiza la gestión del almacenamiento de Amazon FSx for NetApp ONTAP en Kubernetes para que tus desarrolladores y administradores puedan centrarse en la puesta en marcha de aplicaciones. El complemento NetApp Trident EKS incluye los últimos parches de seguridad, correcciones de errores y está validado por AWS para funcionar con Amazon EKS. El complemento EKS te permite garantizar de forma constante que tus clústeres de Amazon EKS sean seguros y estables, y reducir la cantidad de trabajo que necesitas hacer para instalar, configurar y actualizar complementos.

Prerrequisitos

Asegúrate de tener lo siguiente antes de configurar el complemento Trident para AWS EKS:

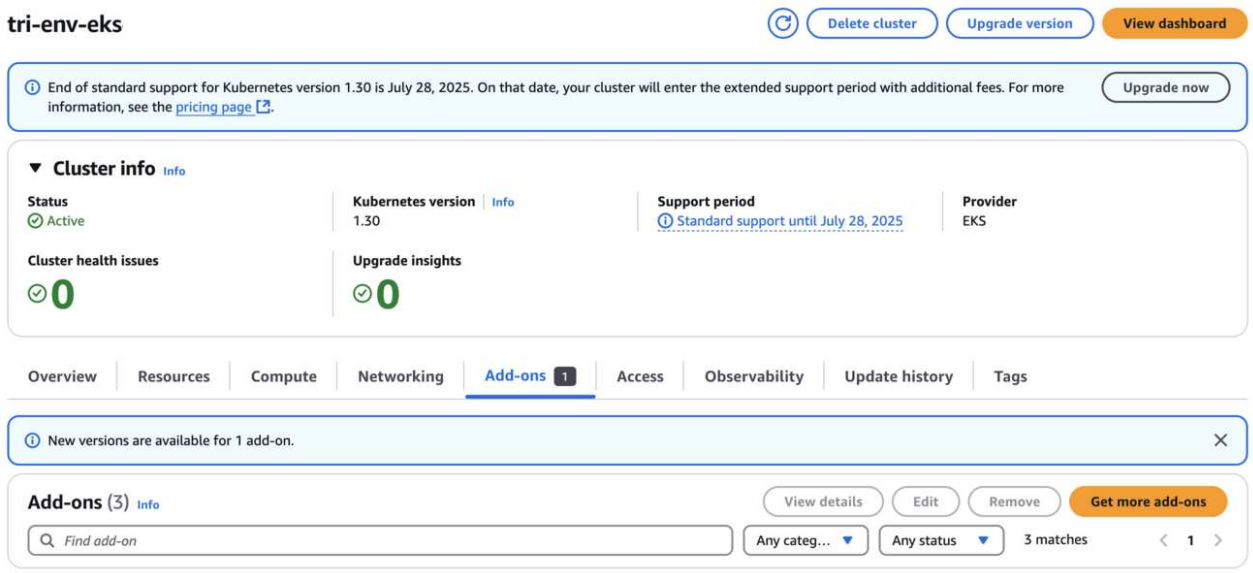
- Una cuenta de clúster de Amazon EKS con permisos para trabajar con complementos. Consulta "Complementos de Amazon EKS".
- Permisos de AWS para AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) o Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nodo: AMD o ARM
- Un sistema de archivos existente de Amazon FSx para NetApp ONTAP

Pasos

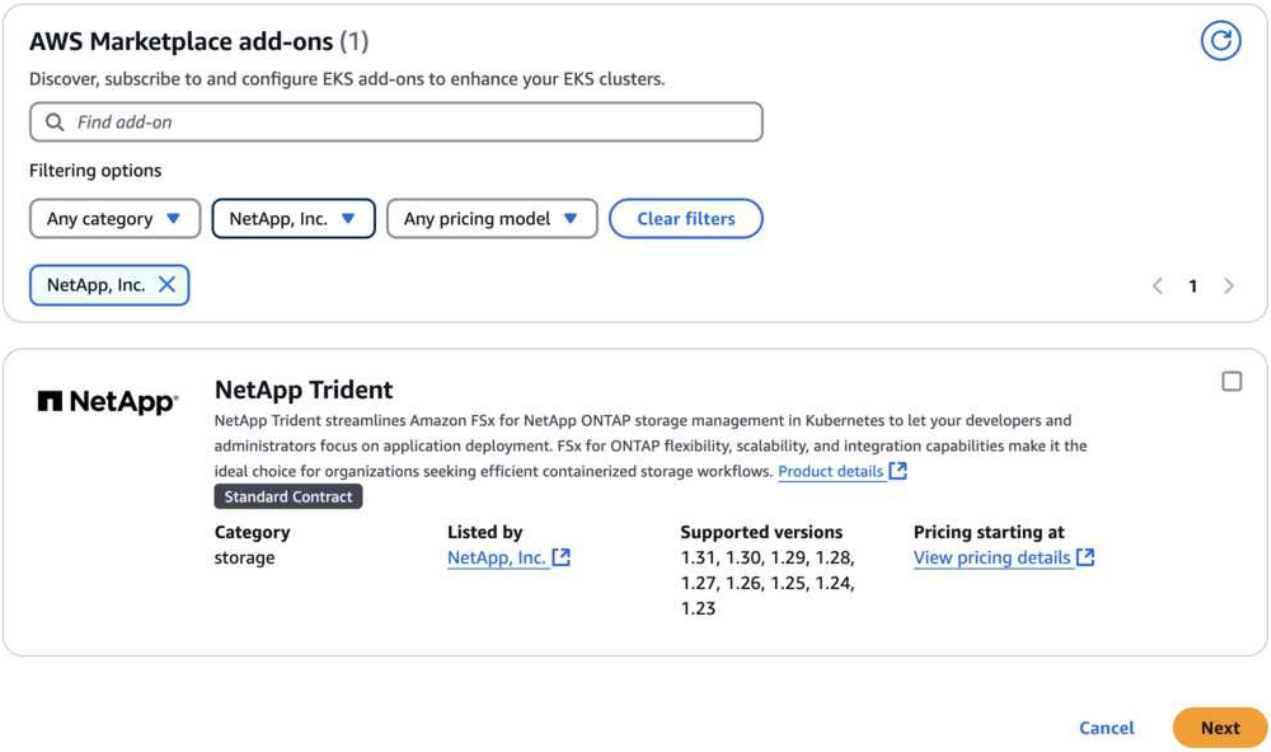
1. Asegúrate de crear un rol de IAM y un secreto de AWS para que los pods de EKS puedan acceder a los

recursos de AWS. Para obtener instrucciones, consulta "Crea un rol de IAM y un secreto de AWS".

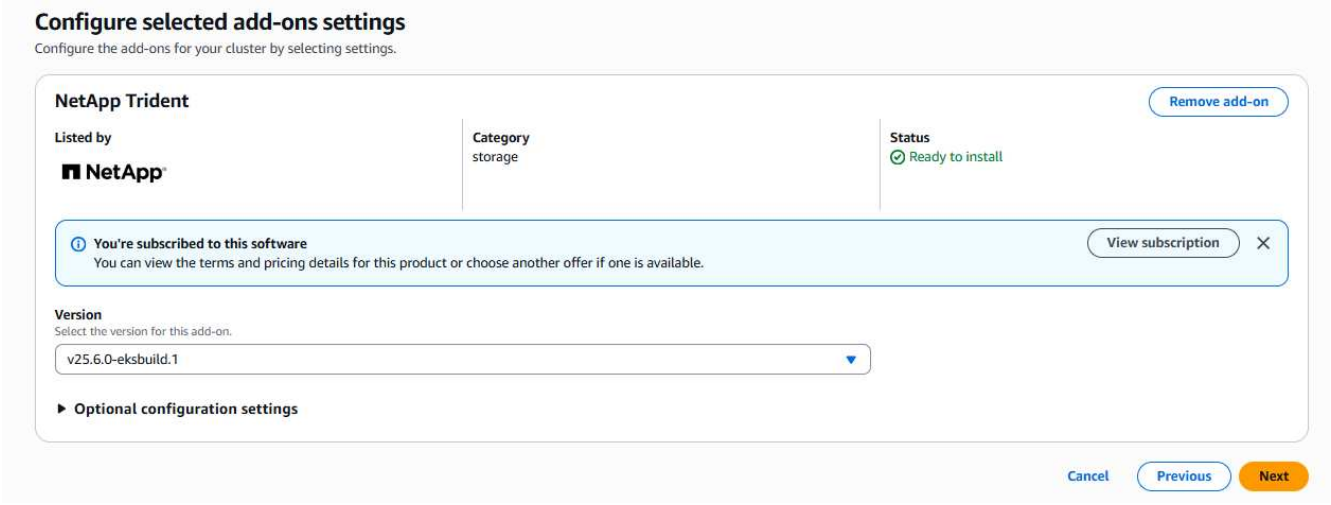
2. En tu clúster EKS Kubernetes, ve a la pestaña **Add-ons**.



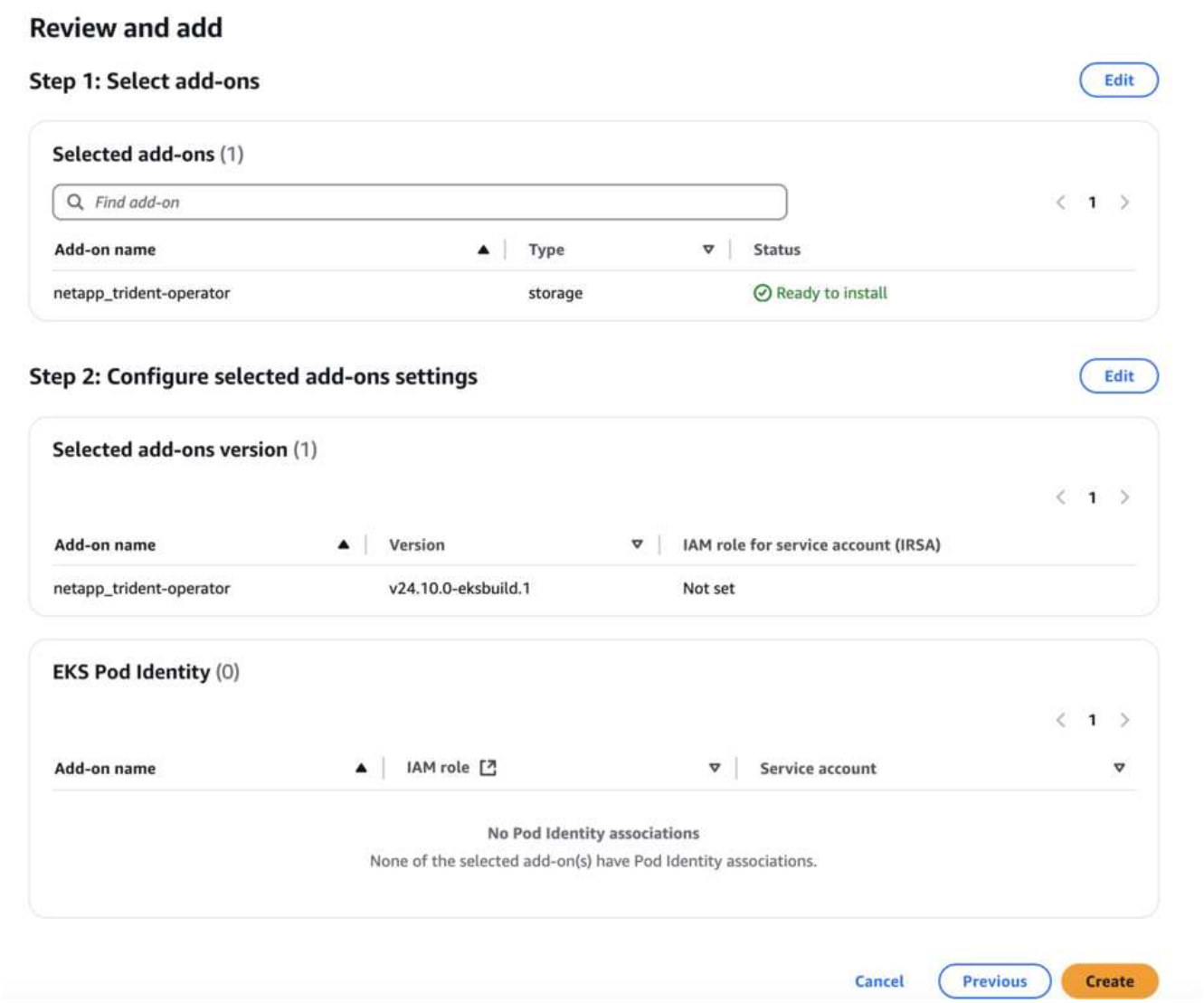
3. Ve a **AWS Marketplace add-ons** y elige la categoría *storage*.



4. Localiza **NetApp Trident** y selecciona la casilla para el complemento Trident, y haz clic en **Siguiente**.
5. Elige la versión deseada del complemento.

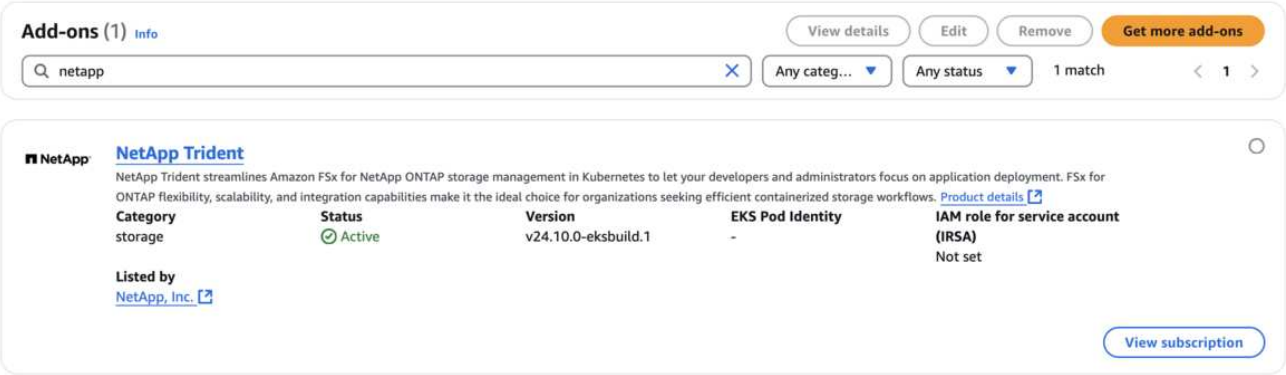


6. Configura los ajustes complementarios necesarios.



7. Si estás usando IRSA (IAM roles para service account), consulta los pasos de configuración adicionales "aquí".
8. Selecciona **Crear**.

9. Verifica que el estado del complemento sea *Activo*.



10. Ejecuta el siguiente comando para verificar que Trident está instalado correctamente en el clúster:

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

11. Continúa con la configuración y configura el backend de almacenamiento. Para más información, consulta "Configura el backend de almacenamiento".

Instala o desinstala el complemento Trident EKS usando la CLI

Instala el complemento Trident EKS de NetApp usando la CLI:

El siguiente comando de ejemplo instala el complemento Trident EKS: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (con una versión dedicada)

El siguiente comando de ejemplo instala el complemento Trident EKS versión 25.6.1: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (con una versión dedicada)

El siguiente comando de ejemplo instala el complemento Trident EKS versión 25.6.2: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (con una versión dedicada)

Desinstala el complemento Trident EKS de NetApp usando la CLI:

El siguiente comando desinstala el complemento Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Crea backends con kubectl

Un backend define la relación entre Trident y un sistema de almacenamiento. Le dice a Trident cómo comunicarse con ese sistema de almacenamiento y cómo Trident debe aprovisionar volúmenes desde él. Después de instalar Trident, el siguiente paso es crear un backend. La `TridentBackendConfig` Custom Resource Definition (CRD) te permite crear y administrar backends de Trident directamente a través de la interfaz de

Kubernetes. Puedes hacer esto usando `kubectl` o la herramienta CLI equivalente para tu distribución de Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) es un CRD frontend con espacio de nombres que te permite administrar backends de Trident usando `kubectl`. Los administradores de Kubernetes y almacenamiento ahora pueden crear y administrar backends directamente a través de la CLI de Kubernetes sin necesitar una utilidad de línea de comandos dedicada (`tridentctl`).

Al crear un `TridentBackendConfig` objeto, sucede lo siguiente:

- Trident crea automáticamente un backend según la configuración que proporciones. Esto se representa internamente como un `TridentBackend` (`tbe`, `tridentbackend`) CR.
- El `TridentBackendConfig` está vinculado de forma única a un `TridentBackend` que fue creado por Trident.

Cada `TridentBackendConfig` mantiene una correspondencia uno a uno con un `TridentBackend`. El primero es la interfaz que se proporciona al usuario para diseñar y configurar backends; el segundo es la forma en que Trident representa el objeto backend real.

> `TridentBackend`Las CR se crean automáticamente por Trident. No deberías modificarlas. Si quieres actualizar los backends, hazlo modificando el objeto `TridentBackendConfig.

Mira el siguiente ejemplo para el formato del `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
```

También puedes echar un vistazo a los ejemplos en el directorio "trident-installer" para ver configuraciones de muestra para la plataforma o servicio de almacenamiento que quieras.

El `spec` toma parámetros de configuración específicos del backend. En este ejemplo, el backend usa el `ontap-san` storage driver y usa los parámetros de configuración que se muestran aquí. Para la lista de opciones de configuración para tu storage driver deseado, consulta el "Información de configuración del backend para tu controlador de almacenamiento".

La `spec`sección también incluye `credentials`y `deletionPolicy`campos, que se introdujeron recientemente en el `TridentBackendConfig`CR:

- `credentials`: Este parámetro es obligatorio y contiene las credenciales utilizadas para autenticarse con el sistema/servicio de almacenamiento. Esto se establece como un secreto de Kubernetes creado por el usuario. Las credenciales no se pueden pasar en texto sin formato y generarán un error.
- `deletionPolicy`: Este campo define qué debe suceder cuando se elimina el `TridentBackendConfig`. Puede tomar uno de dos valores posibles:
  - `delete`: Esto elimina tanto `TridentBackendConfig`CR como el backend asociado. Este es el valor predeterminado.
  - `retain`: Cuando se elimina una `TridentBackendConfig` CR, la definición del backend seguirá presente y se puede administrar con `tridentctl`. Configurar la política de eliminación en `retain` permite a los usuarios volver a una versión anterior (pre-21.04) y conservar los backends creados. El valor de este campo se puede actualizar después de que se cree una `TridentBackendConfig`.

> El nombre de un backend se establece usando `spec.backendName`. Si no se especifica, el nombre del backend se establece como el nombre del objeto `TridentBackendConfig` (metadata.name). Se recomienda establecer explícitamente los nombres de los backends usando `spec.backendName`.

> Los backends que fueron creados con `tridentctl` no tienen un objeto `TridentBackendConfig` asociado. Puedes elegir administrar esos backends con `kubectl` creando un CR `TridentBackendConfig`. Debes tener cuidado de especificar parámetros de configuración idénticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` y así sucesivamente). Trident vinculará automáticamente el `TridentBackendConfig` recién creado con el backend preexistente.

Resumen de pasos

Para crear un nuevo backend usando `kubectl`, debes hacer lo siguiente:

1. Crea un "Kubernetes Secret". El secreto contiene las credenciales que Trident necesita para comunicarse con el clúster/servicio de almacenamiento.
2. Crea un objeto `TridentBackendConfig`. Este contiene información específica sobre el clúster/servicio de almacenamiento y hace referencia al secreto creado en el paso anterior.

Después de crear un backend, puedes observar su estado usando `kubectl get tbc <tbc-name> -n <trident-namespace>` y recopilar detalles adicionales.

Paso 1: crea un secreto de Kubernetes

Crea un secreto que contenga las credenciales de acceso para el backend. Esto es único para cada servicio o plataforma de almacenamiento. Aquí tienes un ejemplo:

```
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
```

Esta tabla resume los campos que deben incluirse en el secreto para cada plataforma de almacenamiento:

Descripción de los campos secretos de la plataforma de almacenamiento	Secreto	Descripción de los campos
Azure NetApp Files	clientID	El ID del cliente de un registro de app
Element (NetApp HCI/SolidFire)	Punto final	MVIP para el clúster SolidFire con credenciales de inquilino
ONTAP	nombre de usuario	Nombre de usuario para conectarse al clúster/SVM. Usado para la autenticación basada en credenciales
ONTAP	contraseña	Contraseña para conectarse al clúster/SVM. Usada para la autenticación basada en credenciales
ONTAP	clientPrivateKey	Valor codificado en Base64 de la clave privada del cliente. Usado para la autenticación basada en certificados
ONTAP	chapUsername	Nombre de usuario entrante. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy`
ONTAP	chapInitiatorSecret	Secreto del iniciador CHAP. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy`

Descripción de los campos secretos de la plataforma de almacenamiento	Secreto	Descripción de los campos
ONTAP	chapTargetUsername	Nombre de usuario de destino. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Secreto del iniciador de destino CHAP. Obligatorio si useCHAP=true. Para `ontap-san` y `ontap-san-economy`

El secreto creado en este paso se va a referenciar en el campo `spec.credentials` del objeto `TridentBackendConfig` que se crea en el siguiente paso.

Paso 2: crea el `TridentBackendConfig`CR

Ya estás listo para crear tu `TridentBackendConfig` CR. En este ejemplo, se crea un backend que usa el `ontap-san` driver mediante el objeto `TridentBackendConfig` que se muestra a continuación:

```
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
```

Paso 3: verifica el estado de la `TridentBackendConfig`CR

Ahora que creaste la `TridentBackendConfig`CR, puedes verificar el estado. Mira el siguiente ejemplo:

```
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
```

Se creó exitosamente un backend y se vinculó al `TridentBackendConfig` CR.

La fase puede tomar uno de los siguientes valores:

- `Bound`: El `TridentBackendConfig` CR está asociado con un backend, y ese backend contiene `configRef` configurado con el `TridentBackendConfig` uid del CR.
- `Unbound`: Representado mediante `""`. El `TridentBackendConfig` objeto no está vinculado a un backend. Todos los nuevos `TridentBackendConfig` CR están en esta fase por defecto. Después de que la fase cambie, no puede volver a estar sin vincular.
- `Deleting`: El `TridentBackendConfig` CR `deletionPolicy` se configuró para eliminarse. Cuando el `TridentBackendConfig` CR se elimina, pasa al estado de eliminación.
  - Si no existen reclamos de volumen persistentes (PVC) en el backend, eliminar el `TridentBackendConfig` hará que Trident elimine el backend así como el `TridentBackendConfig` CR.
  - Si hay una o más PVC en el backend, este pasa a un estado de eliminación. El `TridentBackendConfig` CR posteriormente también entra en fase de eliminación. El backend y `TridentBackendConfig` se eliminan solo después de que se eliminen todas las PVC.
- `Lost`: El backend asociado con el `TridentBackendConfig` CR se eliminó accidental o deliberadamente y el `TridentBackendConfig` CR todavía tiene una referencia al backend eliminado. El `TridentBackendConfig` CR todavía puede eliminarse independientemente del `deletionPolicy` valor.
- `Unknown`: Trident no puede determinar el estado ni la existencia del backend asociado con el `TridentBackendConfig` CR. Por ejemplo, si el servidor API no responde o si el `tridentbackends.trident.netapp.io` CRD falta. Esto podría requerir intervención.

En esta etapa, ¡el backend se ha creado correctamente! Hay varias operaciones que también se pueden gestionar, como "actualizaciones y eliminaciones del backend".

(Opcional) Paso 4: obtén más detalles

Puedes ejecutar el siguiente comando para obtener más información sobre tu backend:

```
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
```

Además, también puedes obtener un volcado YAML/JSON de `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contiene el `backendName` y el `backendUUID` del backend que se creó en respuesta al `TridentBackendConfig` CR. El `lastOperationStatus` campo representa el estado de la última operación del `TridentBackendConfig` CR, que puede ser activada por el usuario (por ejemplo, si el usuario cambió algo en `spec`) o activada por Trident (por ejemplo, durante reinicios de Trident). Puede ser Success o Failed. `phase` representa el estado de la relación entre el `TridentBackendConfig` CR y el backend. En el ejemplo anterior, `phase` tiene el valor Bound, lo que significa que el `TridentBackendConfig` CR está asociado con el backend.

Puedes ejecutar el `kubectl -n trident describe tbc <tbc-cr-name>` comando para obtener detalles de los registros de eventos.

No puedes actualizar ni eliminar un backend que contenga un objeto asociado `TridentBackendConfig` usando `tridentctl`. Para entender los pasos para cambiar entre `tridentctl` y `TridentBackendConfig`, "ver aquí".

Gestiona backends

Realiza la gestión del backend con kubectl

Conoce cómo realizar operaciones de administración de backend usando `kubectl`.

Eliminar un backend

Al eliminar un `TridentBackendConfig`, le indicas a Trident que elimine o conserve los backends (según `deletionPolicy`). Para eliminar un backend, asegúrate de que `deletionPolicy` esté configurado para eliminar. Para eliminar solo el `TridentBackendConfig`, asegúrate de que `deletionPolicy` esté configurado para conservar. Esto asegura que el backend siga presente y se pueda administrar usando `tridentctl`.

Ejecuta el siguiente comando:

```
kubectl delete tbc <tbc-name> -n trident
```

Trident no elimina los secretos de Kubernetes que estaban en uso por `TridentBackendConfig`. El usuario de Kubernetes es responsable de limpiar los secretos. Debes tener cuidado al eliminar secretos. Solo deberías eliminar secretos si no están en uso por los backends.

Ver los backends existentes

Ejecuta el siguiente comando:

```
kubectl get tbc -n trident
```

También puedes ejecutar `tridentctl get backend -n trident` o `tridentctl get backend -o yaml -n trident` para obtener una lista de todos los backends que existen. Esta lista también incluirá los backends que fueron creados con `tridentctl`.

Actualizar un backend

Puede haber varias razones para actualizar un backend:

- Las credenciales del sistema de almacenamiento han cambiado. Para actualizar las credenciales, se debe actualizar el secreto de Kubernetes que se usa en el objeto `TridentBackendConfig`. Trident actualizará automáticamente el backend con las credenciales más recientes proporcionadas. Ejecuta el siguiente comando para actualizar el secreto de Kubernetes:

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Es necesario actualizar los parámetros (como el nombre del ONTAP SVM que se está utilizando).
  - Puedes actualizar `TridentBackendConfig` objetos directamente a través de Kubernetes usando el siguiente comando:

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

  - Alternativamente, puedes hacer cambios en el `TridentBackendConfig`CR existente usando el siguiente comando:

```
kubectl edit tbc <tbc-name> -n trident
```

> - Si falla una actualización del backend, este sigue en su última configuración conocida. Puedes ver los registros para determinar la causa ejecutando `kubectl get tbc <tbc-name> -o yaml -n trident` o `kubectl describe tbc <tbc-name> -n trident`.
> - Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando de actualización.

Realiza la gestión del backend con tridentctl

Conoce cómo realizar operaciones de gestión de backend usando `tridentctl`.

Crear un backend

Después de crear un "archivo de configuración backend", ejecuta el siguiente comando:

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

Si falla la creación del backend, algo estaba mal con la configuración del backend. Puedes ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs -n trident
```

Después de identificar y corregir el problema con el archivo de configuración, puedes simplemente ejecutar el comando `create` otra vez.

Eliminar un backend

Para eliminar un backend de Trident, haz lo siguiente:

1. Recupera el nombre del backend:

```
tridentctl get backend -n trident
```

2. Elimina el backend:

```
tridentctl delete backend <backend-name> -n trident
```

Si Trident ha aprovisionado volúmenes e instantáneas de este backend que aún existen, eliminar el backend impide que se aprovisionen nuevos volúmenes desde él. El backend seguirá existiendo en estado "Eliminando".

Ver los backends existentes

Para ver los backends que Trident conoce, haz lo siguiente:

- Para obtener un resumen, ejecuta el siguiente comando:

```
tridentctl get backend -n trident
```

- Para obtener todos los detalles, ejecuta el siguiente comando:

```
tridentctl get backend -o json -n trident
```

Actualizar un backend

Después de crear un nuevo archivo de configuración de backend, ejecuta el siguiente comando:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Si la actualización del backend falla, algo salió mal con la configuración del backend o intentaste una actualización no válida. Puedes ver los registros para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs -n trident
```

Después de identificar y corregir el problema con el archivo de configuración, simplemente puedes ejecutar el comando `update` otra vez.

Identifica las clases de almacenamiento que usan un backend

Este es un ejemplo del tipo de preguntas que puedes responder con el JSON que `tridentctl` genera para los objetos backend. Esto usa la utilidad `jq`, que necesitas instalar.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Esto también se aplica a los backends que se crearon usando `TridentBackendConfig`.

Moverte entre las opciones de gestión del backend

Conoce las diferentes formas de administrar backends en Trident.

Opciones para gestionar backends

Con la introducción de `TridentBackendConfig`, los administradores ahora tienen dos formas únicas de gestionar los backends. Esto plantea las siguientes preguntas:

- ¿Se pueden gestionar los backends creados usando `tridentctl` con `TridentBackendConfig`?
- ¿Se pueden gestionar los backends creados usando `TridentBackendConfig` mediante `tridentctl`?

Administra `tridentctl` backends usando `TridentBackendConfig`

Esta sección cubre los pasos necesarios para administrar backends que se crearon usando `tridentctl` directamente a través de la interfaz de Kubernetes creando `TridentBackendConfig` objetos.

Esto se aplicará a los siguientes escenarios:

- Backends preexistentes que no tienen un `TridentBackendConfig` porque fueron creados con `tridentctl`.
- Nuevos backends que se crearon con `tridentctl`, mientras que existen otros `TridentBackendConfig` objetos.

En ambos escenarios, los backends seguirán presentes, con Trident programando los volúmenes y operando sobre ellos. Aquí los administradores tienen dos opciones:

- Sigue usando `tridentctl` para administrar los backends que se crearon con él.
- Vincula los backends creados usando `tridentctl` un nuevo `TridentBackendConfig` objeto. Hacer esto significa que los backends se gestionarán usando `kubectl` y no `tridentctl`.

Para administrar un backend preexistente usando `kubectl`, necesitarás crear un `TridentBackendConfig` que se vincule al backend existente. Aquí tienes un resumen de cómo funciona:

1. Crea un secreto de Kubernetes. El secreto contiene las credenciales que Trident necesita para comunicarse con el clúster/servicio de almacenamiento.
2. Crea un `TridentBackendConfig` objeto. Esto contiene detalles sobre el clúster o servicio de almacenamiento y hace referencia al secreto creado en el paso anterior. Debes asegurarte de especificar parámetros de configuración idénticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, y así sucesivamente). `spec.backendName` debe establecerse con el nombre del backend existente.

Paso 0: identifica el backend

Para crear un `TridentBackendConfig` que se vincule a un backend existente, necesitas obtener la configuración del backend. En este ejemplo, supongamos que se creó un backend usando la siguiente definición JSON:

```
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"
      }
    }
  ]
}
```

Paso 1: crea un secreto de Kubernetes

Crea un secreto que contenga las credenciales para el backend, como se muestra en este ejemplo:

```
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
```

Paso 2: crea un `TridentBackendConfig`CR

El siguiente paso es crear una `TridentBackendConfig` CR que se vincule automáticamente con la preexistente `ontap-nas-backend` (como en este ejemplo). Asegúrate de que se cumplan los siguientes requisitos:

- El mismo nombre de backend se define en `spec.backendName`.
- Los parámetros de configuración son idénticos al backend original.
- Los pools virtuales (si están presentes) deben conservar el mismo orden que en el backend original.
- Las credenciales se proporcionan a través de un Kubernetes Secret y no en texto sin formato.

En este caso, el `TridentBackendConfig` se verá así:

```
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
```

Paso 3: verifica el estado de la `TridentBackendConfig`CR

Después de que el `TridentBackendConfig` haya sido creado, su fase debe ser `Bound`. También debe reflejar el mismo nombre de backend y UUID que el backend existente.

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

Ahora el backend se va a administrar completamente usando el `tbc-ontap-nas-backend TridentBackendConfig` objeto.

Administra `TridentBackendConfig`backends usando `tridentctl`

```
`tridentctl` se puede usar para listar los backends que fueron creados
usando  `TridentBackendConfig`. Además, los administradores también pueden
elegir administrar completamente dichos backends a través de  `tridentctl`
eliminando  `TridentBackendConfig` y asegurándose de que
`spec.deletionPolicy` esté configurado como  `retain`.
```

Paso 0: identifica el backend

Por ejemplo, supongamos que el siguiente backend se creó usando `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Desde la salida, se puede ver que `TridentBackendConfig` fue creado exitosamente y está vinculado a un backend [observa el UUID del backend].

Paso 1: confirma que `deletionPolicy` esté configurado en `retain`

Echemos un vistazo al valor de `deletionPolicy`. Esto debe configurarse en `retain`. Esto asegura que cuando se elimina una CR de `TridentBackendConfig`, la definición del backend seguirá presente y podrás gestionarla con `tridentctl`.

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

No sigas con el siguiente paso a menos que `deletionPolicy` esté configurado en `retain`.

Paso 2: elimina el `TridentBackendConfig` CR

El último paso es eliminar el `TridentBackendConfig` CR. Después de confirmar que el `deletionPolicy` está configurado como `retain`, puedes continuar con la eliminación:

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

Al eliminar el `TridentBackendConfig` objeto, Trident simplemente lo elimina sin eliminar realmente el backend en sí.

Crea y gestiona clases de almacenamiento

Crear una clase de almacenamiento

Configura un objeto StorageClass de Kubernetes y crea la clase de almacenamiento para indicarle a Trident cómo aprovisionar volúmenes.

Configura un objeto StorageClass de Kubernetes

El "Objeto de Kubernetes StorageClass" identifica a Trident como el aprovisionador que se usa para esa clase y le indica a Trident cómo aprovisionar un volumen. Por ejemplo:

```
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
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

Crear una clase de almacenamiento

Después de crear el objeto StorageClass, puedes crear la clase de almacenamiento. Ejemplos de storage class proporciona algunos ejemplos básicos que puedes usar o modificar.

Pasos

1. Este es un objeto de Kubernetes, así que usa `kubectl` para crearlo en Kubernetes.

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

2. Ahora deberías ver una clase de almacenamiento **basic-csi** tanto en Kubernetes como en Trident, y Trident debería haber descubierto los pools en el backend.

```
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"
        ]
      }
    }
  ]
}
```

Ejemplos de storage class

Trident proporciona "definiciones de clases de almacenamiento simples para backends específicos".

Alternativamente, puedes editar `sample-input/storage-class-csi.yaml.templ` archivo que viene con el instalador y reemplazar *`BACKEND_TYPE`* con el nombre del controlador de almacenamiento.

```
./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
```

Administra clases de almacenamiento

Puedes ver las clases de almacenamiento existentes, establecer una clase de almacenamiento predeterminada, identificar el backend de la clase de almacenamiento y eliminar clases de almacenamiento.

Ver las clases de almacenamiento existentes

- Para ver las clases de almacenamiento de Kubernetes existentes, ejecuta el siguiente comando:

```
kubectl get storageclass
```

- Para ver los detalles de la clase de almacenamiento de Kubernetes, ejecuta el siguiente comando:

```
kubectl get storageclass <storage-class> -o json
```

- Para ver las clases de almacenamiento sincronizadas de Trident, ejecuta el siguiente comando:

```
tridentctl get storageclass
```

- Para ver los detalles de la clase de almacenamiento sincronizado de Trident, ejecuta el siguiente comando:

```
tridentctl get storageclass <storage-class> -o json
```

Establece una clase de almacenamiento predeterminada

Kubernetes 1.6 agregó la capacidad de establecer una clase de almacenamiento predeterminada. Esta es la clase de almacenamiento que se usará para aprovisionar un volumen persistente si un usuario no especifica una en una reclamación de volumen persistente (PVC).

- Define una clase de almacenamiento predeterminada estableciendo la anotación `storageclass.kubernetes.io/is-default-class` como true en la definición de la clase de almacenamiento. Según la especificación, cualquier otro valor o la ausencia de la anotación se interpreta como false.
- Puedes configurar una clase de almacenamiento existente para que sea la clase de almacenamiento predeterminada usando el siguiente comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- De manera similar, puedes eliminar la anotación de clase de almacenamiento predeterminada usando el siguiente comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

También hay ejemplos en el instalador de Trident que incluyen esta anotación.

> Solo debe haber una clase de almacenamiento predeterminada en tu clúster a la vez. Kubernetes no impide técnicamente que tengas más de una, pero se comportará como si no hubiera ninguna clase de almacenamiento predeterminada.

Identifica el backend para una clase de almacenamiento

Este es un ejemplo del tipo de preguntas que puedes responder con el JSON que `tridentctl` genera para los objetos backend de Trident. Esto usa la utilidad `jq`, que puede que necesites instalar primero.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Eliminar una clase de almacenamiento

Para eliminar una clase de almacenamiento de Kubernetes, ejecuta el siguiente comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` debe reemplazarse con tu clase de almacenamiento.

Cualquier volumen persistente que se haya creado a través de esta clase de almacenamiento permanecerá intacto, y Trident seguirá administrándolos.

> Trident aplica un espacio en blanco `fsType` para los volúmenes que crea. Para los backends iSCSI, se recomienda aplicarlo `parameters.fsType` en el StorageClass. Deberías eliminar los StorageClasses existentes y volver a crearlos con `parameters.fsType` especificado.

Aprovisiona y gestiona volúmenes

Aprovisiona un volumen

Crea un PersistentVolumeClaim (PVC) que use el StorageClass de Kubernetes configurado para solicitar acceso al PV. Luego puedes montar el PV en un pod.

Descripción general

Una "*PersistentVolumeClaim*" (PVC) es una solicitud de acceso al PersistentVolume en el clúster.

El PVC se puede configurar para solicitar almacenamiento de un tamaño o modo de acceso determinados. Usando el StorageClass asociado, el administrador del clúster puede controlar más que solo el tamaño y el modo de acceso de PersistentVolume, como el rendimiento o el nivel de servicio.

Después de crear el PVC, puedes montar el volumen en un pod.

Crea el PVC

Pasos

1. Crea el PVC.

```
kubectl create -f pvc.yaml
```

2. Verifica el estado del PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Monta el volumen en un pod.

```
kubectl create -f pv-pod.yaml
```

Puedes monitorear el progreso usando `kubectl get pod --watch`.

2. Verifica que el volumen esté montado en `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Ahora puedes eliminar el Pod. La aplicación del Pod ya no existirá, pero el volumen permanecerá.

```
kubectl delete pod pv-pod
```

Manifiestos de muestra

Manifiestos de muestra de PersistentVolumeClaim

Estos ejemplos muestran opciones básicas de configuración de PVC.

PVC con acceso RWO

Este ejemplo muestra un PVC básico con acceso RWO que está asociado con una StorageClass llamada `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC con NVMe/TCP

Este ejemplo muestra un PVC básico para NVMe/TCP con acceso RWO que está asociado con una StorageClass llamada `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Ejemplos de manifiestos de Pod

Estos ejemplos muestran configuraciones básicas para conectar el PVC a un pod.

Configuración básica

```
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
```

Configuración básica de NVMe/TCP

```
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
```

Consulta "Objetos de Kubernetes y Trident" para ver detalles sobre cómo las clases de almacenamiento interactúan con el `PersistentVolumeClaim` y los parámetros para controlar cómo Trident aprovisiona volúmenes.

Ampliar volúmenes

Trident permite a los usuarios de Kubernetes ampliar sus volúmenes después de que se crean. Encuentra información sobre las configuraciones necesarias para ampliar volúmenes iSCSI, NFS, SMB, NVMe/TCP y FC.

Expandir un volumen iSCSI

Puedes expandir un volumen persistente iSCSI (PV) usando el aprovisionador CSI.

> La expansión de volumen iSCSI es compatible con los `ontap-san`, `ontap-san-economy`, `solidfire-san` controladores y requiere Kubernetes 1.16 y versiones posteriores.

Paso 1: Configura la StorageClass para admitir la expansión de volumen

Edita la definición de StorageClass para establecer el campo `allowVolumeExpansion` en `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para un StorageClass ya existente, edítalo para incluir el `allowVolumeExpansion` parámetro.

Paso 2: Crea un PVC con el StorageClass que creaste

Edita la definición de PVC y actualiza el `spec.resources.requests.storage` para reflejar el nuevo tamaño deseado, que debe ser mayor que el tamaño original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crea un volumen persistente (PV) y lo asocia con este Persistent Volume Claim (PVC).

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Paso 3: define un pod que adjunta el PVC

Conecta el PV a un pod para que pueda ser redimensionado. Hay dos escenarios cuando redimensionas un PV iSCSI:

- Si el PV está conectado a un pod, Trident expande el volumen en el backend de almacenamiento, vuelve a escanear el dispositivo y redimensiona el sistema de archivos.
- Al intentar redimensionar un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Después de que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Luego, Kubernetes actualiza el tamaño del PVC después de que la operación de expansión se haya completado correctamente.

En este ejemplo, se crea un pod que usa el `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Paso 4: expandir el PV

Para cambiar el tamaño del PV que se ha creado de 1Gi a 2Gi, edita la definición de PVC y actualiza el `spec.resources.requests.storage` a 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
 # ...
```

Paso 5: valida la expansión

Puedes validar que la expansión funcionó correctamente verificando el tamaño del PVC, PV y el volumen de Trident:

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

Expandir un volumen FC

Puedes expandir un volumen persistente (PV) FC usando el aprovisionador CSI.

> La expansión del volumen FC es compatible con el `ontap-san` driver y requiere Kubernetes 1.16 y versiones posteriores.

Paso 1: Configura la StorageClass para admitir la expansión de volumen

Edita la definición de StorageClass para establecer el campo `allowVolumeExpansion` en `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para un StorageClass ya existente, edítalo para incluir el `allowVolumeExpansion` parámetro.

Paso 2: Crea un PVC con el StorageClass que creaste

Edita la definición de PVC y actualiza el `spec.resources.requests.storage` para reflejar el nuevo tamaño deseado, que debe ser mayor que el tamaño original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crea un volumen persistente (PV) y lo asocia con este Persistent Volume Claim (PVC).

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Paso 3: define un pod que adjunta el PVC

Conecta el PV a un pod para que pueda ser redimensionado. Hay dos escenarios cuando se redimensiona un PV de FC:

- Si el PV está conectado a un pod, Trident expande el volumen en el backend de almacenamiento, vuelve a escanear el dispositivo y redimensiona el sistema de archivos.
- Al intentar redimensionar un PV no conectado, Trident expande el volumen en el backend de almacenamiento. Después de que el PVC se vincula a un pod, Trident vuelve a escanear el dispositivo y redimensiona el sistema de archivos. Luego, Kubernetes actualiza el tamaño del PVC después de que la

operación de expansión se haya completado correctamente.

En este ejemplo, se crea un pod que usa el `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Paso 4: expandir el PV

Para cambiar el tamaño del PV que se ha creado de 1Gi a 2Gi, edita la definición de PVC y actualiza el `spec.resources.requests.storage` a 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
 # ...
```

Paso 5: valida la expansión

Puedes validar que la expansión funcionó correctamente verificando el tamaño del PVC, PV y el volumen de Trident:

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

Expandir un volumen NFS

Trident admite la expansión de volumen para PVs NFS aprovisionados en `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` y `azure-netapp-files` backends.

Paso 1: Configura la StorageClass para admitir la expansión de volumen

Para cambiar el tamaño de un PV NFS, el administrador primero debe configurar la clase de almacenamiento para permitir la expansión del volumen configurando el campo `allowVolumeExpansion` en `true`:

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si ya creaste una clase de almacenamiento sin esta opción, simplemente puedes editar la clase de

almacenamiento existente usando `kubectl edit storageclass` para permitir la expansión del volumen.

Paso 2: Crea un PVC con el StorageClass que creaste

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident debería crear un NFS PV de 20 MiB para este PVC:

```
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
```

Paso 3: expandir el PV

Para cambiar el tamaño del PV de 20 MiB recién creado a 1 GiB, edita el PVC y establece `spec.resources.requests.storage` en 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
# ...
```

Paso 4: valida la expansión

Puedes validar que el cambio de tamaño funcionó correctamente verificando el tamaño del PVC, PV y el volumen Trident:

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

Importar volúmenes

Puedes importar volúmenes de almacenamiento existentes como un PV de Kubernetes usando `tridentctl import` o creando una Persistent Volume Claim (PVC) con anotaciones de importación de Trident.

Descripción general y consideraciones

Puedes importar un volumen en Trident para:

- Containerizar una aplicación y reutilizar su conjunto de datos existente
- Usa un clon de un conjunto de datos para una aplicación efímera
- Reconstruye un clúster de Kubernetes fallido
- Migra datos de aplicaciones durante la recuperación ante desastres

Consideraciones

Antes de importar un volumen, revisa las siguientes consideraciones.

- Trident solo puede importar volúmenes ONTAP de tipo RW (lectura-escritura). Los volúmenes de tipo DP (protección de datos) son volúmenes destino de SnapMirror. Debes romper la relación de espejo antes de

importar el volumen en Trident.

- Sugerimos importar volúmenes sin conexiones activas. Para importar un volumen en uso activo, clona el volumen y luego haz la importación.

> Esto es especialmente importante para los volúmenes de bloques, ya que Kubernetes no sabría de la conexión anterior y podría conectar fácilmente un volumen activo a un pod. Esto puede provocar la corrupción de datos.

- Aunque `StorageClass` debe especificarse en una PVC, Trident no usa este parámetro durante la importación. Las clases de almacenamiento se usan durante la creación del volumen para seleccionar entre los pools disponibles según las características del almacenamiento. Como el volumen ya existe, no se requiere seleccionar un pool durante la importación. Así que la importación no fallará incluso si el volumen existe en un backend o pool que no coincide con la clase de almacenamiento especificada en la PVC.
- El tamaño del volumen existente se determina y se configura en el PVC. Después de que el controlador de almacenamiento importa el volumen, el PV se crea con un ClaimRef al PVC.
  - La política de recuperación se establece inicialmente en `retain` el PV. Después de que Kubernetes vincula correctamente el PVC y el PV, la política de recuperación se actualiza para que coincida con la política de recuperación de la Storage Class.
  - Si la política de recuperación de la Storage Class es `delete`, el volumen de almacenamiento se eliminará cuando se elimine el PV.
- De forma predeterminada, Trident administra el PVC y renombra el volumen FlexVol y el LUN en el backend. Puedes pasar la `--no-manage` bandera para importar un volumen no administrado y la `--no-rename` bandera para conservar el nombre del volumen.
  - `--no-manage`* - Si usas la `--no-manage`flag, Trident no realiza ninguna operación adicional en el PVC ni en el PV durante el ciclo de vida de los objetos. El volumen de almacenamiento no se elimina cuando se elimina el PV y otras operaciones como clonar el volumen y redimensionar el volumen también se ignoran.
  - `--no-rename`* - Si usas la `--no-rename`flag, Trident conserva el nombre del volumen existente al importar volúmenes y administra el ciclo de vida de los volúmenes. Esta opción solo es compatible con los `ontap-nas`, `ontap-san` (incluidos los sistemas ASA r2), y `ontap-san-economy`drivers.

> Estas opciones son útiles si quieres usar Kubernetes para cargas de trabajo en contenedores, pero si prefieres, puedes gestionar el ciclo de vida del volumen de almacenamiento fuera de Kubernetes.

- Se añade una anotación al PVC y al PV que cumple una doble función: indicar que el volumen fue importado y si el PVC y el PV están gestionados. Esta anotación no debe modificarse ni eliminarse.

Importar un volumen

Puedes importar un volumen usando `tridentctl import` o creando un PVC con anotaciones de importación de Trident.

> Si usas anotaciones de PVC, no necesitas descargar ni usar `tridentctl` para importar el volumen.

Usando tridentctl

Pasos

1. Crea un archivo PVC (por ejemplo, `pvc.yaml`) que se usará para crear el PVC. El archivo PVC debe incluir `name`, `namespace`, `accessModes` y `storageClassName`. Opcionalmente, puedes especificar `unixPermissions` en tu definición de PVC.

   El siguiente es un ejemplo de una especificación mínima:

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

   > Incluye solo los parámetros necesarios. Parámetros adicionales como el nombre de PV o el tamaño del volumen pueden hacer que falle el comando de import.

2. Usa el `tridentctl import` comando para especificar el nombre del backend de Trident que contiene el volumen y el nombre que identifica de forma única el volumen en el almacenamiento (por ejemplo: ONTAP FlexVol, Element Volume). El argumento `-f` es necesario para especificar la ruta al archivo PVC.

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

Uso de anotaciones de PVC

Pasos

1. Crea un archivo PVC YAML (por ejemplo, `pvc.yaml`) con las anotaciones de importación de Trident necesarias. El archivo PVC debe incluir:

   - `name` y `namespace` en metadatos
   - `accessModes`, `resources.requests.storage` y `storageClassName` en spec
   - Anotaciones:
     - `trident.netapp.io/importOriginalName`: nombre del volumen en el backend
     - `trident.netapp.io/importBackendUUID`: UUID del backend donde existe el volumen
     - `trident.netapp.io/notManaged` (*Opcional*): configúralo en `"true"` para volúmenes no gestionados. El valor predeterminado es `"false"`.

       El siguiente es un ejemplo de especificación para importar un volumen gestionado:

```
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. Aplica el archivo YAML de PVC a tu clúster de Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automáticamente el volumen y lo vinculará al PVC.

Ejemplos

Revisa los siguientes ejemplos de importación de volúmenes para los controladores compatibles.

ONTAP NAS y ONTAP NAS FlexGroup

Trident admite la importación de volúmenes usando los `ontap-nas` y `ontap-nas-flexgroup` controladores.

- Trident no admite la importación de volúmenes usando el `ontap-nas-economy` driver.
- Los `ontap-nas` y `ontap-nas-flexgroup` controladores no permiten nombres de volúmenes duplicados.

Cada volumen creado con el `ontap-nas` controlador es un volumen FlexVol en el clúster ONTAP. Importar volúmenes FlexVol con el `ontap-nas` controlador funciona igual. Un volumen FlexVol que ya existe en un clúster ONTAP se puede importar como un `ontap-nas` PVC. De manera similar, los volúmenes FlexGroup se pueden importar como `ontap-nas-flexgroup` PVCs.

Ejemplos de ONTAP NAS usando tridentctl

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados usando `tridentctl`.

Volumen administrado

El siguiente ejemplo importa un volumen denominado `managed_volume` en un backend denominado `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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volumen no administrado

Al usar el `--no-manage` argumento, Trident no cambia el nombre del volumen.

El siguiente ejemplo importa `unmanaged_volume` en el `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   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Ejemplos de ONTAP NAS usando anotaciones de PVC

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados usando anotaciones de PVC.

Volumen administrado

El siguiente ejemplo importa un volumen de 1GiB `ontap-nas` llamado `ontap_volume1` desde backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` con el modo de acceso RWO establecido usando anotaciones PVC:

```
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>
```

Volumen no administrado

El siguiente ejemplo importa 1Gi `ontap-nas` volumen llamado `ontap-volume2` desde backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` con el modo de acceso RWO configurado usando anotaciones PVC:

```
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 admite la importación de volúmenes usando los `ontap-san` (iSCSI, NVMe/TCP y FC) y `ontap-san-economy` drivers.

Trident puede importar volúmenes SAN de ONTAP FlexVol que contienen un solo LUN. Esto es coherente con el `ontap-san` driver, que crea un volumen FlexVol para cada PVC y un LUN dentro del volumen FlexVol. Trident importa el volumen FlexVol y lo asocia con la definición de PVC. Trident puede importar volúmenes `ontap-san-economy` que contienen varios LUN.

Los siguientes ejemplos muestran cómo importar volúmenes gestionados y no gestionados:

Volumen administrado

Para los volúmenes administrados, Trident cambia el nombre del FlexVol volumen al `pvc-<uuid>` formato y el LUN dentro del FlexVol volumen a `lun0`.

El siguiente ejemplo importa el `ontap-san-managed` volumen FlexVol que está presente en el `ontap_san_default` backend:

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

Volumen no administrado

El siguiente ejemplo importa `unmanaged_example_volume` en el `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   |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Si tienes LUN asignados a igroups que comparten un IQN con un IQN de nodo Kubernetes, como se muestra en el siguiente ejemplo, recibirás el error: `LUN already mapped to initiator(s) in this group`. Necesitarás eliminar el iniciador o desasignar el LUN para importar el volumen.

```
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
```

Elemento

Trident admite el software NetApp Element y la importación de volúmenes NetApp HCI usando el controlador `solidfire-san`.

> El controlador Element admite nombres de volumen duplicados. Sin embargo, Trident devuelve un error si hay nombres de volumen duplicados. Como solución alternativa, clona el volumen, ponle un nombre de volumen único e importa el volumen clonado.

El siguiente ejemplo importa un `element-managed` volumen en el 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 admite la importación de volúmenes usando el controlador `azure-netapp-files`.

> Para importar un volumen de Azure NetApp Files, identifica el volumen por su ruta de volumen. La ruta de volumen es la parte de la ruta de exportación del volumen después de `:/`. Por ejemplo, si la ruta de montaje es `10.0.0.2:/importvol1`, la ruta de volumen es `importvol1`.

El siguiente ejemplo importa un `azure-netapp-files` volumen en el backend `azurenetappfiles_40517` con la ruta de volumen `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 admite la importación de volúmenes usando el `google-cloud-netapp-volumes` driver.

El siguiente ejemplo importa un volumen en backend `backend-tbc-gcnv1` con el volumen `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
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

El siguiente ejemplo importa un `google-cloud-netapp-volumes` volumen cuando hay dos volúmenes presentes en la misma región:

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

Personaliza los nombres y las etiquetas de los volúmenes

Con Trident, puedes asignar nombres y etiquetas significativos a los volúmenes que creas. Esto te ayuda a identificar y asignar fácilmente los volúmenes a sus respectivos recursos de Kubernetes (PVCs). También puedes definir plantillas en el backend para crear nombres y etiquetas de volumen personalizados; cualquier volumen que crees, importes o clones se ajustará a las plantillas.

Antes de empezar

Soporte para nombres y etiquetas de volumen personalizables:

- Operaciones de crear, importar y clonar volúmenes.
- En el caso del `ontap-nas-economy`driver, solo el nombre del volumen Qtree cumple con la plantilla de nombre.
- En el caso del `ontap-san-economy`driver, solo el nombre LUN cumple con la plantilla de nombre.

Limitaciones

- Los nombres de volúmenes personalizados solo son compatibles con los drivers on-premises de ONTAP.
- Las etiquetas personalizadas solo son compatibles con los `ontap-san`, `ontap-nas` y `ontap-nas-flexgroup` controladores.
- Los nombres de volúmenes personalizados no se aplican a los volúmenes existentes.

Comportamientos clave de los nombres de volúmenes personalizables

- Si se produce un error debido a una sintaxis no válida en una plantilla de nombre, la creación del backend

falla. Sin embargo, si la aplicación de la plantilla falla, el volumen se nombrará según la convención de nomenclatura existente.

- El prefijo de almacenamiento no es aplicable cuando un volumen se nombra usando una plantilla de nombres de la configuración del backend. Cualquier valor de prefijo que quieras se puede agregar directamente a la plantilla.

Ejemplos de configuración de backend con plantilla de nombre y etiquetas

Se pueden definir plantillas de nombres personalizadas en el nivel raíz y/o en el nivel de pool.

Ejemplo de nivel raíz

```
{
  "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}}"
  }
}
```

Ejemplo a nivel de pool

```
{
  "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 }}"
      }
    }
  ]
}
```

Ejemplos de plantillas de nombres

Ejemplo 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Ejemplo 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Puntos a considerar

1. En el caso de las importaciones de volumen, las etiquetas se actualizan solo si el volumen existente tiene etiquetas en un formato específico. Por ejemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. En el caso de importaciones de volúmenes administrados, el nombre del volumen sigue la plantilla de nombre definida en el nivel raíz en la definición del backend.
3. Trident no admite el uso de un operador de corte con el prefijo de almacenamiento.
4. Si las plantillas no dan como resultado nombres de volumen únicos, Trident agregará algunos caracteres aleatorios para crear nombres de volumen únicos.
5. Si el nombre personalizado de un volumen NAS economy supera los 64 caracteres, Trident nombrará los volúmenes según la convención de nomenclatura existente. Para todos los demás controladores ONTAP, si el nombre del volumen supera el límite de caracteres, el proceso de creación del volumen falla.

Comparte un volumen NFS entre espacios de nombres

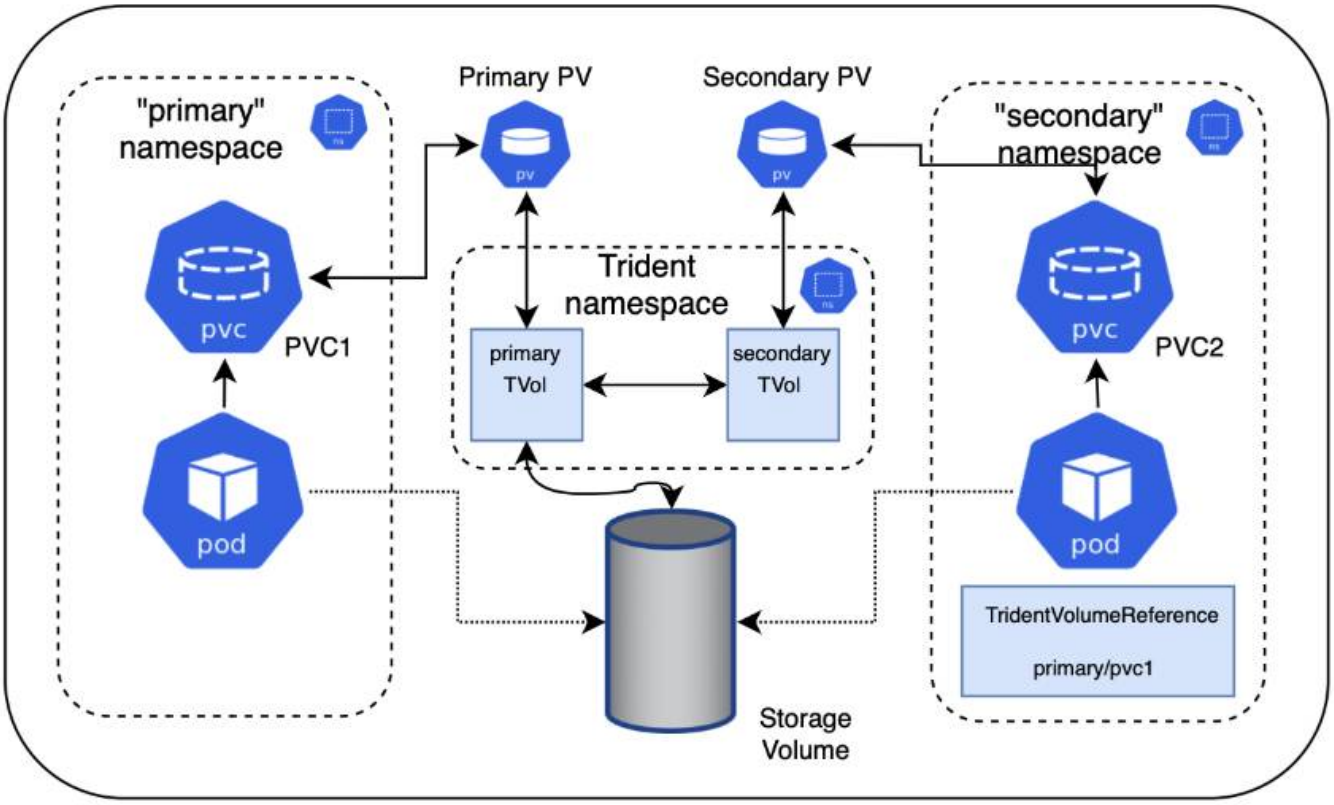
Con Trident, puedes crear un volumen en un espacio de nombres primario y compartirlo en uno o más espacios de nombres secundarios.

Funciones

El CR TridentVolumeReference te permite compartir de forma segura volúmenes NFS ReadWriteMany (RWX) entre uno o varios espacios de nombres de Kubernetes. Esta solución nativa de Kubernetes tiene los siguientes beneficios:

- Múltiples niveles de control de acceso para garantizar la seguridad
- Funciona con todos los controladores de volumen NFS de Trident
- No depende de tridentctl ni de ninguna otra función no nativa de Kubernetes

Este diagrama ilustra el uso compartido de volúmenes NFS en dos espacios de nombres de Kubernetes.



Inicio rápido

Puedes configurar el uso compartido de volúmenes NFS en unos pocos pasos.

1 **Configura el PVC de origen para compartir el volumen**

El propietario del espacio de nombres de origen otorga permiso para acceder a los datos en el PVC de origen.

2 **Otorga permiso para crear una CR en el espacio de nombres de destino**

El administrador del clúster otorga permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference.

3 **Crea TridentVolumeReference en el espacio de nombres de destino**

El propietario del espacio de nombres de destino crea el CR TridentVolumeReference para hacer referencia al PVC de origen.

4 **Crea la PVC subordinada en el espacio de nombres de destino**

El propietario del espacio de nombres de destino crea el PVC subordinado para usar la fuente de datos del PVC de origen.

Configura los espacios de nombres de origen y destino

Para garantizar la seguridad, el uso compartido entre espacios de nombres requiere la colaboración y acción del propietario del espacio de nombres de origen, el administrador del clúster y el propietario del espacio de nombres de destino. El rol del usuario se designa en cada paso.

Pasos

1. **Propietario del espacio de nombres de origen:** Crea el PVC (`pvc1`) en el espacio de nombres de origen que da permiso para compartir con el espacio de nombres de destino (`namespace2`) usando la anotación `shareToNamespace`.

```
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 crea el PV y su volumen de almacenamiento NFS de backend.

   - Puedes compartir el PVC con varios espacios de nombres usando una lista separada por comas. Por ejemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Puedes compartir a todos los espacios de nombres usando `*`. Por ejemplo, `trident.netapp.io/shareToNamespace: *`
   - Puedes actualizar el PVC para incluir la anotación `shareToNamespace` en cualquier momento.

2. **Administrador del clúster:** Asegúrate de que existe el RBAC adecuado para conceder permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference en el espacio de nombres de destino.
3. **Propietario del espacio de nombres de destino:** crea una TridentVolumeReference CR en el espacio de nombres de destino que hace referencia al espacio de nombres de origen `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propietario del espacio de nombres de destino:** crea un PVC (`pvc2`) en el espacio de nombres de destino (`namespace2`) usando la anotación `shareFromPVC` para designar el PVC de origen.

```
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
```

El tamaño del PVC de destino debe ser menor o igual que el PVC de origen.

Resultados

Trident lee la `shareFromPVC` anotación en el PVC de destino y crea el PV de destino como un volumen subordinado sin recurso de almacenamiento propio que apunta al PV de origen y comparte el recurso de almacenamiento del PV de origen. El PVC y el PV de destino aparecen vinculados de forma normal.

Eliminar un volumen compartido

Puedes eliminar un volumen que está compartido entre varios espacios de nombres. Trident quitará el acceso al volumen en el espacio de nombres de origen y mantendrá el acceso para los demás espacios de nombres que comparten el volumen. Cuando se eliminan todos los espacios de nombres que hacen referencia al volumen, Trident elimina el volumen.

Usa `tridentctl get` para consultar volúmenes subordinados

Usando la utilidad[`tridentctl`, puedes ejecutar el comando `get` para obtener volúmenes subordinados. Para más información, consulta el enlace:../trident-reference/trident-cl.html[`tridentctl` comandos y

opciones].

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

Banderas:

- `-h, --help`: ayuda para volúmenes.
- `--parentOfSubordinate string`: limitar la consulta al volumen fuente subordinado.
- `--subordinateOf string`: limitar la consulta a los subordinados del volumen.

Limitaciones

- Trident no puede evitar que los espacios de nombres de destino escriban en el volumen compartido. Deberías usar el bloqueo de archivos u otros procesos para evitar sobrescribir los datos del volumen compartido.
- No puedes revocar el acceso al PVC de origen eliminando las anotaciones `shareToNamespace` o `shareFromNamespace` o eliminando el `TridentVolumeReference` CR. Para revocar el acceso, tienes que borrar el PVC subordinado.
- Las instantáneas, clones y la replicación no son posibles en volúmenes subordinados.

Para más información

Para obtener más información sobre el acceso a volúmenes entre espacios de nombres:

- Visita "Compartir volúmenes entre espacios de nombres: di hola al acceso a volúmenes entre espacios de nombres".
- Mira la demo en "NetAppTV".

Clonar volúmenes en distintos espacios de nombres

Con Trident, puedes crear nuevos volúmenes usando volúmenes existentes o instantáneas de volúmenes de un espacio de nombres diferente dentro del mismo clúster de Kubernetes.

Prerrequisitos

Antes de clonar volúmenes, asegúrate de que los backends de origen y destino sean del mismo tipo y tengan la misma clase de almacenamiento.

> La clonación entre espacios de nombres solo se admite para los controladores de almacenamiento `ontap-san` y `ontap-nas`. No se admiten clones de solo lectura.

Inicio rápido

Puedes configurar la clonación de volumen en solo unos pocos pasos.

1 Configura el PVC de origen para clonar el volumen

El propietario del espacio de nombres de origen otorga permiso para acceder a los datos en el PVC de origen.

2 Otorga permiso para crear una CR en el espacio de nombres de destino

El administrador del clúster otorga permiso al propietario del espacio de nombres de destino para crear el CR TridentVolumeReference.

3 Crea TridentVolumeReference en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea el CR TridentVolumeReference para hacer referencia al PVC de origen.

4 Crea el PVC clonado en el espacio de nombres de destino

El propietario del espacio de nombres de destino crea PVC para clonar la PVC del espacio de nombres de origen.

Configura los espacios de nombres de origen y destino

Para garantizar la seguridad, clonar volúmenes entre espacios de nombres requiere la colaboración y acción del propietario del espacio de nombres de origen, el administrador del clúster y el propietario del espacio de nombres de destino. El rol del usuario se indica en cada paso.

Pasos

1. **Propietario del espacio de nombres de origen:** Crea el PVC (`pvc1` en el espacio de nombres de origen (`namespace1` que otorga permiso para compartir con el espacio de nombres de destino (`namespace2` usando la anotación `cloneToNamespace`.

```
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 crea el PV y su volumen de almacenamiento backend.

- Puedes compartir la PVC con varios espacios de nombres usando una lista delimitada por comas. Por ejemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Puedes compartir con todos los espacios de nombres usando `*`. Por ejemplo, `trident.netapp.io/cloneToNamespace: *`
- Puedes actualizar el PVC para incluir la `cloneToNamespace` anotación en cualquier momento.

2. **Administrador del clúster:** Asegúrate de que exista un RBAC adecuado para otorgar permiso al propietario del espacio de nombres de destino para crear la CR TridentVolumeReference en el espacio de nombres de destino (`namespace2`).
3. **Propietario del espacio de nombres de destino:** crea una TridentVolumeReference CR en el espacio de nombres de destino que hace referencia al espacio de nombres de origen `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propietario del espacio de nombres de destino:** crea un PVC (`pvc2`) en el espacio de nombres de destino (`namespace2`) usando la `cloneFromPVC` o `cloneFromSnapshot`, y `cloneFromNamespace` anotaciones para designar el PVC de origen.

```
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
```

Limitaciones

- Para los PVC aprovisionados usando controladores ontap-nas-economy, no se admiten clones de solo lectura.

Replica volúmenes usando SnapMirror

Trident admite relaciones de espejo entre un volumen de origen en un clúster y el volumen de destino en el clúster emparejado para replicar datos para recuperación ante desastres. Puedes usar una Custom Resource Definition (CRD) con espacio de nombres, llamada Trident Mirror Relationship (TMR), para realizar las siguientes operaciones:

- Crear relaciones de espejo entre volúmenes (PVCs)
- Eliminar relaciones de espejo entre volúmenes
- Rompe las relaciones espejo
- Promociona el volumen secundario durante condiciones de desastre (conmutaciones por error)
- Realiza una transición sin pérdidas de aplicaciones de un clúster a otro (durante conmutaciones por error o migraciones planificadas)

Requisitos previos de replicación

Asegúrate de que se cumplan los siguientes requisitos previos antes de comenzar:

Clústeres ONTAP

- **Trident**: Trident versión 22.10 o posterior debe existir tanto en los clústeres de Kubernetes de origen como en los de destino que utilizan ONTAP como backend.
- **Licencias**: Las licencias asíncronas de ONTAP SnapMirror usando el paquete Data Protection deben estar habilitadas tanto en el clúster ONTAP de origen como en el clúster de destino. Consulta "Resumen de licencias de SnapMirror en ONTAP" para más información.

  A partir de ONTAP 9.10.1, todas las licencias se entregan como un archivo de licencia NetApp (NLF), que es un único archivo que habilita varias funciones. Consulta "Licencias incluidas con ONTAP One" para más información.

  > Solo se admite la protección asíncrona de SnapMirror.

Emparejamiento

- **Clúster y SVM**: Los backends de almacenamiento de ONTAP deben estar interconectados. Consulta "Descripción general de clúster y SVM peering" para más información.

  > Asegúrate de que los nombres de SVM usados en la relación de replicación entre dos clústeres ONTAP sean únicos.

- **Trident y SVM**: las SVM remotas emparejadas deben estar disponibles para Trident en el clúster de destino.

Controladores compatibles

NetApp Trident admite la replicación de volúmenes con la tecnología NetApp SnapMirror usando clases de

almacenamiento respaldadas por los siguientes controladores: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (requiere como mínimo la versión ONTAP 9.15.1)

> La replicación de volúmenes usando SnapMirror no es compatible con sistemas ASA r2. Para información sobre los sistemas ASA r2, consulta "Conoce los sistemas de almacenamiento ASA r2".

Crear un PVC reflejado

Sigue estos pasos y usa los ejemplos de CRD para crear una relación de espejo entre los volúmenes primario y secundario.

Pasos

1. Realiza los siguientes pasos en el clúster principal de Kubernetes:
   a. Crea un objeto StorageClass con el parámetro `trident.netapp.io/replication: true`.

      Ejemplo

      ```
      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. Crea un PVC con una StorageClass creada previamente.

      Ejemplo

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Crea un CR de MirrorRelationship con información local.

Ejemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident obtiene la información interna del volumen y el estado actual de protección de datos (DP) del volumen, luego rellena el campo de estado de MirrorRelationship.

d. Obtén el TridentMirrorRelationship CR para obtener el nombre interno y SVM del PVC.

```
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. Realiza los siguientes pasos en el clúster secundario de Kubernetes:

   a. Crea un StorageClass con el parámetro trident.netapp.io/replication: true.

Ejemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Crea un CR MirrorRelationship con información sobre el destino y el origen.

Ejemplo

```
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 creará una relación SnapMirror con el nombre de política de relación configurado (o el predeterminado para ONTAP) y la inicializará.

c. Crea un PVC con el StorageClass creado previamente para que actúe como secundario (SnapMirror destino).

Ejemplo

```
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 comprobará si existe el TridentMirrorRelationship CRD y fallará al crear el volumen si la relación no existe. Si la relación existe, Trident se asegurará de que el nuevo volumen FlexVol se coloque en una SVM que esté peered con la SVM remota definida en el MirrorRelationship.

Estados de replicación de volúmenes

Una Trident Mirror Relationship (TMR) es un CRD que representa un extremo de una relación de replicación entre PVCs. La TMR de destino tiene un estado, que le dice a Trident cuál es el estado deseado. La TMR de destino tiene los siguientes estados:

- **Establecido**: el PVC local es el volumen de destino de una relación de espejo, y esta es una nueva relación.
- **Promocionado**: el PVC local es ReadWrite y montable, sin relación de espejo actualmente en vigor.
- **Reestablecido**: el PVC local es el volumen de destino de una relación de espejo y también estaba previamente en esa relación de espejo.
  - El estado restablecido debe usarse si el volumen de destino estuvo alguna vez en una relación con el volumen de origen porque sobrescribe el contenido del volumen de destino.
  - El estado restablecido fallará si el volumen no estaba previamente en una relación con el origen.

Promociona el PVC secundario durante una conmutación por error no planificada

Realiza el siguiente paso en el clúster secundario de Kubernetes:

- Actualiza el campo *spec.state* de TridentMirrorRelationship a `promoted`.

Promociona el PVC secundario durante una conmutación por error planificada

Durante una conmutación por error planificada (migración), realiza los siguientes pasos para promover el PVC secundario:

Pasos

1. En el clúster primario de Kubernetes, crea una instantánea del PVC y espera hasta que se cree la instantánea.
2. En el clúster primario de Kubernetes, crea el CR SnapshotInfo para obtener detalles internos.

   Ejemplo

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. En el clúster de Kubernetes secundario, actualiza el campo *spec.state* del CR *TridentMirrorRelationship* a *promoted* y *spec.promotedSnapshotHandle* al internalName de la instantánea.
4. En el clúster secundario de Kubernetes, confirma el estado (campo status.state) de TridentMirrorRelationship a promoted.

Restaura una relación de réplica tras una conmutación por error

Antes de restablecer una relación de espejo, elige el lado que quieres convertir en el nuevo primario.

Pasos

1. En el clúster secundario de Kubernetes, asegúrate de que los valores del campo *spec.remoteVolumeHandle* en el TridentMirrorRelationship estén actualizados.
2. En el clúster de Kubernetes secundario, actualiza el campo *spec.mirror* de TridentMirrorRelationship a `reestablished`.

Operaciones adicionales

Trident admite las siguientes operaciones en los volúmenes primario y secundario:

Replica el PVC primario en un nuevo PVC secundario

Asegúrate de que ya tienes un PVC primario y un PVC secundario.

Pasos

1. Elimina los PersistentVolumeClaim y TridentMirrorRelationship CRD del clúster secundario (de destino) establecido.
2. Elimina el CRD TridentMirrorRelationship del clúster principal (origen).
3. Crea un nuevo CRD TridentMirrorRelationship en el clúster principal (de origen) para el nuevo PVC secundario (de destino) que quieres establecer.

Redimensiona un PVC reflejado, primario o secundario

El PVC se puede redimensionar de forma normal, ONTAP expandirá automáticamente cualquier destino flevxols si la cantidad de datos excede el tamaño actual.

Eliminar la replicación de un PVC

Para eliminar la replicación, realiza una de las siguientes operaciones en el volumen secundario actual:

- Elimina el MirrorRelationship en el PVC secundario. Esto rompe la relación de replicación.
- O, actualiza el campo spec.state a *promoted*.

Elimina un PVC (que antes estaba replicado)

Trident comprueba si hay PVC replicados y libera la relación de replicación antes de intentar eliminar el volumen.

Eliminar un TMR

Eliminar una TMR en un lado de una relación de réplica hace que la TMR restante pase al estado *promoted* antes de que Trident complete la eliminación. Si la TMR seleccionada para eliminar ya está en estado *promoted*, no existe una relación de réplica y la TMR se eliminará y Trident promoverá el PVC local a *ReadWrite*. Esta eliminación libera los metadatos de SnapMirror para el volumen local en ONTAP. Si este volumen se usa en una relación de réplica en el futuro, debe usar una nueva TMR con un estado de replicación de volumen *established* al crear la nueva relación de réplica.

Actualizar las relaciones de réplica cuando ONTAP está en línea

Las relaciones de espejo se pueden actualizar en cualquier momento después de que se establecen. Puedes usar los campos `state: promoted` o `state: reestablished` para actualizar las relaciones. Al promover un volumen de destino a un volumen normal ReadWrite, puedes usar *promotedSnapshotHandle* para especificar una instantánea específica a la que restaurar el volumen actual.

Actualiza las relaciones de réplica cuando ONTAP está fuera de línea

Puedes usar un CRD para realizar una actualización de SnapMirror sin que Trident tenga conectividad directa con el clúster ONTAP. Consulta el siguiente formato de ejemplo de TridentActionMirrorUpdate:

Ejemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` refleja el estado del TridentActionMirrorUpdate CRD. Puede tomar un valor de *Succeeded*, *In Progress* o *Failed*.

Usa la topología CSI

Trident puede crear y adjuntar volúmenes de forma selectiva a los nodos presentes en un clúster de Kubernetes mediante el uso de "Función de topología CSI".

Descripción general

Con la función CSI Topology, el acceso a los volúmenes se puede limitar a un subconjunto de nodos, según las regiones y zonas de disponibilidad. Hoy en día, los proveedores de nube permiten a los administradores de Kubernetes crear nodos que están basados en zonas. Los nodos pueden estar ubicados en diferentes zonas de disponibilidad dentro de una región o en varias regiones. Para facilitar el aprovisionamiento de volúmenes para cargas de trabajo en una arquitectura multizona, Trident utiliza CSI Topology.

Conoce más sobre la función CSI Topology "aquí".

Kubernetes ofrece dos modos de enlace de volumen únicos:

- Con `VolumeBindingMode` configurado en `Immediate`, Trident crea el volumen sin ninguna conciencia de topología. La vinculación de volúmenes y el aprovisionamiento dinámico se gestionan cuando se crea la PVC. Esta es la opción predeterminada `VolumeBindingMode` y es adecuada para clústeres que no aplican restricciones de topología. Los volúmenes persistentes se crean sin tener ninguna dependencia de los requisitos de programación del pod solicitante.
- Con `VolumeBindingMode` establecido en `WaitForFirstConsumer`, la creación y vinculación de un volumen persistente para una PVC se retrasa hasta que se programa y crea un pod que usa la PVC. De esta forma, se crean volúmenes para cumplir con las restricciones de programación impuestas por los requisitos de topología.

El `WaitForFirstConsumer`modo de enlace no requiere etiquetas de topología. Esto puede usarse independientemente de la función de topología CSI.

Lo que necesitas

Para utilizar la topología CSI, necesitas lo siguiente:

- Un clúster de Kubernetes ejecutando un "versión de Kubernetes compatible"

```
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"}
```

- Los nodos del clúster deben tener etiquetas que permitan reconocer la topología (`topology.kubernetes.io/region` y `topology.kubernetes.io/zone`). Estas etiquetas **deberían estar presentes en los nodos del clúster** antes de instalar Trident para que Trident reconozca la topología.

```
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"}]
```

Paso 1: crea un backend que tenga en cuenta la topología

Los backends de almacenamiento Trident pueden diseñarse para aprovisionar volúmenes selectivamente según las zonas de disponibilidad. Cada backend puede incluir un `supportedTopologies` bloque opcional que representa una lista de zonas y regiones compatibles. Para StorageClasses que hacen uso de dicho backend, solo se creará un volumen si lo solicita una aplicación programada en una región o zona compatible.

Aquí tienes un ejemplo de definición de backend:

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"
    }
  ]
}
```

`supportedTopologies` se utiliza para proporcionar una lista de regiones y zonas por backend. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en un StorageClass. Para StorageClasses que contienen un subconjunto de las regiones y zonas proporcionadas en un backend, Trident crea un volumen en el backend.

Puedes definir `supportedTopologies` por grupo de almacenamiento también. Mira el siguiente ejemplo:

```
---
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
```

En este ejemplo, las `region` y `zone` etiquetas representan la ubicación del grupo de almacenamiento. `topology.kubernetes.io/region` y `topology.kubernetes.io/zone` indican desde dónde se pueden consumir los grupos de almacenamiento.

Paso 2: Define StorageClasses que son conscientes de la topología

Según las etiquetas de topología que se proporcionan a los nodos en el clúster, se pueden definir StorageClasses para que contengan información de topología. Esto determinará los grupos de almacenamiento que sirven como candidatos para las solicitudes de PVC realizadas y el subconjunto de nodos que pueden usar los volúmenes proporcionados por Trident.

Mira el siguiente ejemplo:

```
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
```

En la definición de StorageClass proporcionada arriba, `volumeBindingMode` se establece en `WaitForFirstConsumer`. Las PVC que se soliciten con este StorageClass no se procesarán hasta que se haga referencia a ellas en un pod. Y, `allowedTopologies` proporciona las zonas y la región que se van a usar. El `netapp-san-us-east1` StorageClass crea PVC en el backend definido arriba `san-backend-us-east1`.

Paso 3: crea y utiliza un PVC

Con el StorageClass creado y asignado a un backend, ahora puedes crear PVCs.

Mira el ejemplo `spec` abajo:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Crear un PVC usando este manifiesto daría como resultado lo siguiente:

```
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
```

Para que Trident cree un volumen y lo vincule al PVC, usa el PVC en un pod. Mira el siguiente ejemplo:

```yaml
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
```

Este podSpec le indica a Kubernetes que programe el pod en los nodos que están presentes en la `us-east1` región y que elija entre cualquier nodo que esté presente en las `us-east1-a` o `us-east1-b` zonas.

Mira el siguiente resultado:

```
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
```

Actualizar los backends para incluir `supportedTopologies`

Los backends preexistentes se pueden actualizar para incluir una lista de `supportedTopologies` usando `tridentctl backend update`. Esto no afectará a los volúmenes que ya se hayan aprovisionado y solo se usará para PVC posteriores.

Encuentra más información

- "Administra recursos para contenedores"
- "nodeSelector"
- "Afinidad y antiafinidad"
- "Taints y tolerations"

Trabaja con instantáneas

Las instantáneas de volúmenes de Kubernetes de Persistent Volumes (PVs) permiten copias en un momento específico de los volúmenes. Puedes crear una instantánea de un volumen creado usando Trident, importar una instantánea creada fuera de Trident, crear un nuevo volumen a partir de una instantánea existente y recuperar datos de volúmenes desde instantáneas.

Descripción general

La instantánea de volumen es compatible con `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` y `google-cloud-netapp-volumes` controladores.

Antes de empezar

Debes tener un controlador de instantáneas externo y definiciones de recursos personalizadas (CRDs) para trabajar con instantáneas. Esto es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

Si tu distribución de Kubernetes no incluye el controlador de instantáneas y los CRD, consulta Implementa un controlador de instantáneas de volumen.

No crees un controlador de instantáneas si vas a crear instantáneas de volumen bajo demanda en un entorno GKE. GKE usa un controlador de instantáneas integrado y oculto.

Crear una instantánea de volumen

Pasos

1. Crea un `VolumeSnapshotClass`. Para más información, consulta "VolumeSnapshotClass".
   - El `driver` apunta al controlador Trident CSI.
   - `deletionPolicy` puede ser `Delete` o `Retain`. Cuando se establece en `Retain`, la instantánea física subyacente en el clúster de almacenamiento se conserva incluso cuando el objeto `VolumeSnapshot` se elimina.

     Ejemplo

     ```
     cat snap-sc.yaml
     ```

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

2. Crea una instantánea de un PVC existente.

   Ejemplos

   - Este ejemplo crea una instantánea de un PVC existente.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Este ejemplo crea un objeto de instantánea de volumen para un PVC llamado `pvc1` y el nombre de la instantánea se establece en `pvc1-snap`. Un VolumeSnapshot es análogo a un PVC y está asociado con un `VolumeSnapshotContent` objeto que representa la instantánea real.

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

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

- Puedes identificar el objeto `VolumeSnapshotContent` para el `pvc1-snap` VolumeSnapshot describiéndolo. El `Snapshot Content Name` identifica el objeto VolumeSnapshotContent que sirve a esta instantánea. El parámetro `Ready To Use` indica que la instantánea puede usarse para crear un nuevo PVC.

```
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
...
```

Crear una PVC a partir de una instantánea de volumen

Puedes usar `dataSource` para crear un PVC usando un VolumeSnapshot llamado `<pvc-name>` como fuente de los datos. Después de crear el PVC, puedes adjuntarlo a un pod y usarlo como cualquier otro PVC.

> El PVC se creará en el mismo backend que el volumen de origen. Consulta "KB: no se puede crear un PVC a partir de una instantánea de PVC de Trident en un backend alternativo".

El siguiente ejemplo crea el PVC usando `pvc1-snap` como fuente de datos.

```
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
```

Importar una instantánea de volumen

Trident admite la "Proceso de instantáneas preaprovisionadas de Kubernetes" para que el administrador del clúster pueda crear un `VolumeSnapshotContent` objeto e importar instantáneas creadas fuera de Trident.

Antes de empezar

Trident debe haber creado o importado el volumen primario de la instantánea.

Pasos

1. **Administrador del clúster:** Crea un `VolumeSnapshotContent` objeto que hace referencia a la instantánea del backend. Esto inicia el flujo de trabajo de la instantánea en Trident.
   - Especifica el nombre de la instantánea del backend en `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especifica `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` en `snapshotHandle`. Esta es la única información proporcionada a Trident por el snapshotter externo en la llamada `ListSnapshots`.

     > El `<volumeSnapshotContentName>` no siempre puede coincidir con el nombre de la instantánea de backend debido a las restricciones de nomenclatura de CR.

     Ejemplo

     El siguiente ejemplo crea un `VolumeSnapshotContent` objeto que hace referencia a la instantánea del backend `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Cluster admin:** Crea el `VolumeSnapshot` CR que hace referencia al `VolumeSnapshotContent` objeto. Esto solicita acceso para usar el `VolumeSnapshot` en un espacio de nombres determinado.

   Ejemplo

   El siguiente ejemplo crea un `VolumeSnapshot` CR llamado `import-snap` que hace referencia al `VolumeSnapshotContent` llamado `import-snap-content`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. **Procesamiento interno (no requiere acción):** el snapshotter externo reconoce el recién creado `VolumeSnapshotContent` y ejecuta la llamada `ListSnapshots`. Trident crea el `TridentSnapshot`.
   - El external snapshotter establece el `VolumeSnapshotContent` en `readyToUse` y el `VolumeSnapshot` en `true`.
   - Trident devuelve `readyToUse=true`.
4. **Cualquier usuario:** Crea un `PersistentVolumeClaim` para hacer referencia al nuevo `VolumeSnapshot`, donde el `spec.dataSource` (o `spec.dataSourceRef`) nombre es el nombre de `VolumeSnapshot`.

Ejemplo

El siguiente ejemplo crea una PVC que hace referencia al `VolumeSnapshot` llamado `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recupera datos de volumen usando instantáneas

El directorio de instantáneas está oculto de forma predeterminada para facilitar la máxima compatibilidad de los volúmenes aprovisionados usando los `ontap-nas` y `ontap-nas-economy` controladores. Habilita el directorio `.snapshot` para recuperar datos directamente de las instantáneas.

Usa la CLI de ONTAP para restaurar una instantánea de volumen y devolver un volumen a un estado registrado en una instantánea anterior.

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

> Cuando restauras una copia instantánea, la configuración existente del volumen se sobrescribe. Los cambios realizados en los datos del volumen después de que se creó la copia instantánea se pierden.

Restauración de volumen en el lugar desde una instantánea

Trident permite la restauración rápida e in situ de volúmenes a partir de una instantánea mediante la `TridentActionSnapshotRestore` (TASR) CR. Esta CR funciona como una acción imperativa de Kubernetes y no persiste después de que la operación se completa.

Trident admite la restauración de instantáneas en los `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` y `solidfire-san` controladores.

Antes de empezar

Debes tener un PVC enlazado y una instantánea de volumen disponible.

- Verifica que el estado del PVC esté enlazado.

```
kubectl get pvc
```

- Verifica que la instantánea de volumen esté lista para usar.

```
kubectl get vs
```

Pasos

1. Crea la CR de TASR. Este ejemplo crea una CR para PVC `pvc1` y volumen snapshot `pvc1-snapshot`.

   El TASR CR debe estar en un espacio de nombres donde existan el PVC y el VS.

```
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. Aplica la CR para restaurar desde la instantánea. Este ejemplo restaura desde la instantánea `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

Trident restaura los datos de la instantánea. Puedes verificar el estado de la restauración de la instantánea:

```
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: ""
```

- En la mayoría de los casos, Trident no reintentará automáticamente la operación en caso de fallo. Tendrás que realizar la operación de nuevo.
- Es posible que los usuarios de Kubernetes sin acceso de administrador tengan que recibir permiso del administrador para crear un TASR CR en el espacio de nombres de su aplicación.

Eliminar un PV con instantáneas asociadas

Al eliminar un volumen persistente con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un "estado Eliminando". Elimina las instantáneas del volumen para eliminar el volumen Trident.

Implementa un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el snapshot controller y los CRDs, puedes implementarlos así.

Pasos

1. Crea CRD de instantáneas de volumen.

```
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. Crea el controlador de instantáneas.

```
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
```

Si es necesario, abre `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` y actualiza `namespace` a tu espacio de nombres.

Enlaces relacionados

- "Instantáneas de volumen"
- "VolumeSnapshotClass"

Trabaja con instantáneas de grupo de volúmenes

Instantáneas de grupos de volúmenes de Kubernetes de volúmenes persistentes (PVs) NetApp Trident te permite crear instantáneas de varios volúmenes (un grupo de instantáneas de volúmenes). Esta instantánea de grupo de volúmenes representa copias de varios volúmenes tomadas en el mismo momento específico.

VolumeGroupSnapshot es una función beta en Kubernetes con API beta. Kubernetes 1.32 es la versión mínima requerida para VolumeGroupSnapshot.

Crear instantáneas de grupo de volúmenes

La instantánea del grupo de volúmenes es compatible con los siguientes controladores de almacenamiento:

- `ontap-san`controlador: solo para los protocolos iSCSI y FC, no para el protocolo NVMe/TCP.
- `ontap-san-economy` - solo para el protocolo iSCSI.
- `ontap-nas`

> La instantánea del grupo de volúmenes no es compatible con NetApp ASA r2 o los sistemas de almacenamiento AFX.

Antes de empezar

- Asegúrate de que tu versión de Kubernetes sea K8s 1.32 o superior.
- Debes tener un controlador de instantáneas externo y definiciones de recursos personalizadas (CRDs) para trabajar con instantáneas. Esto es responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

  Si tu distribución de Kubernetes no incluye el controlador de instantáneas externo y los CRDs, consulta Implementa un controlador de instantáneas de volumen.

  > No crees un controlador de instantáneas si vas a crear instantáneas de grupo de volúmenes bajo demanda en un entorno de GKE. GKE usa un controlador de instantáneas integrado y oculto.

- En el YAML del controlador de instantáneas, configura la `CSIVolumeGroupSnapshot` feature gate en 'true' para asegurarte de que la instantánea del grupo de volúmenes esté habilitada.
- Crea las clases de instantáneas de grupo de volúmenes necesarias antes de crear una instantánea de grupo de volúmenes.
- Asegúrate de que todos los PVC/volúmenes estén en el mismo SVM para poder crear VolumeGroupSnapshot.

Pasos

- Crea un VolumeGroupSnapshotClass antes de crear un VolumeGroupSnapshot. Para más información, consulta "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
```

- Crea PVC con las etiquetas requeridas usando clases de almacenamiento existentes o agrega estas etiquetas a los PVC existentes.

  El siguiente ejemplo crea la PVC usando `pvc1-group-snap` como origen de datos y la etiqueta

`consistentGroupSnapshot: groupA`. Define la clave y el valor de la etiqueta según tus necesidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crea un VolumeGroupSnapshot con la misma etiqueta (`consistentGroupSnapshot: groupA`) especificada en el PVC.

  Este ejemplo crea una instantánea de grupo de volúmenes:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recupera datos de volumen usando una instantánea de grupo

Puedes restaurar volúmenes persistentes individuales usando las instantáneas individuales que se han creado como parte de la instantánea del grupo de volúmenes. No puedes recuperar la instantánea del grupo de volúmenes como una unidad.

Usa la CLI de ONTAP para restaurar una instantánea de volumen y devolver un volumen a un estado registrado en una instantánea anterior.

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

Cuando restauras una copia instantánea, la configuración existente del volumen se sobrescribe. Los cambios realizados en los datos del volumen después de que se creó la copia instantánea se pierden.

Restauración de volumen en el lugar desde una instantánea

Trident permite la restauración rápida e in situ de volúmenes a partir de una instantánea mediante la `TridentActionSnapshotRestore` (TASR) CR. Esta CR funciona como una acción imperativa de Kubernetes y no persiste después de que la operación se completa.

Para más información, consulta "Restauración de volumen en el lugar desde una instantánea".

Eliminar un PV con instantáneas de grupo asociadas

Al eliminar una instantánea de grupo de volúmenes:

- Puedes eliminar VolumeGroupSnapshots como un todo, no instantáneas individuales en el grupo.
- Si se eliminan PersistentVolumes mientras existe una instantánea para ese PersistentVolume, Trident moverá ese volumen a un estado de "eliminación" porque la instantánea debe eliminarse antes de que el volumen pueda eliminarse de manera segura.
- Si se ha creado un clon utilizando una instantánea agrupada y luego se desea eliminar el grupo, comenzará una operación de división en clon y el grupo no se puede eliminar hasta que se complete la división.

Implementa un controlador de instantáneas de volumen

Si tu distribución de Kubernetes no incluye el snapshot controller y los CRDs, puedes implementarlos así.

Pasos

1. Crea CRD de instantáneas de volumen.

```
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. Crea el controlador de instantáneas.

```
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
```

Si es necesario, abre `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` y actualiza `namespace` a tu espacio de nombres.

Enlaces relacionados

- "VolumeGroupSnapshotClass"
- "Instantáneas de volumen"

Administra y supervisa Trident

Actualizar Trident

Actualizar Trident

A partir de la versión 24.02, Trident sigue un ritmo de lanzamiento de cuatro meses, entregando tres versiones principales cada año calendario. Cada nueva versión se basa en las versiones anteriores y ofrece nuevas funciones, mejoras de rendimiento, correcciones de errores y mejoras. Te recomendamos actualizar al menos una vez al año para aprovechar las nuevas funciones de Trident.

Consideraciones antes de actualizar

Al actualizar a la última versión de Trident, ten en cuenta lo siguiente:

- Solo debe haber una instancia de Trident instalada en todos los namespaces en un clúster de Kubernetes dado.
- Trident 23.07 y versiones posteriores requieren instantáneas de volumen v1 y ya no son compatibles con instantáneas alfa o beta.
- Al actualizar, es importante que proporciones `parameter.fsType` en `StorageClasses` usados por Trident. Puedes eliminar y volver a crear `StorageClasses` sin interrumpir los volúmenes preexistentes.
  - Este es un **requisito** para aplicar "contextos de seguridad" a los volúmenes SAN.
  - El directorio sample input contiene ejemplos, como `storage-class-basic.yaml.templ` y `storage-class-bronze-default.yaml`.
  - Para más información, consulta "Problemas conocidos".

Paso 1: selecciona una versión

Las versiones de Trident siguen una `YY.MM` convención de nomenclatura basada en la fecha, donde "YY" son los dos últimos dígitos del año y "MM" es el mes. Las versiones puntuales siguen una `YY.MM.X` convención, donde "X" es el nivel del parche. Vas a seleccionar la versión a la que quieres actualizar según la versión desde la que estés actualizando.

- Puedes hacer una actualización directa a cualquier versión de destino que esté dentro de una ventana de cuatro versiones respecto a la versión que tienes instalada. Por ejemplo, puedes actualizar directamente de 24.06 (o cualquier versión puntual de 24.06) a 25.06.
- Si estás actualizando desde una versión fuera del periodo de cuatro versiones, haz una actualización en varios pasos. Usa las instrucciones de actualización para la "versión anterior" de la que estás actualizando para pasar a la versión más reciente que entre en el periodo de cuatro versiones. Por ejemplo, si estás usando la versión 23.07 y quieres actualizar a la 25.06:
  a. Primera actualización de 23.07 a 24.06.
  b. Luego actualiza de 24.06 a 25.06.

Cuando actualices usando el operador Trident en OpenShift Container Platform, deberías actualizar a Trident 21.01.1 o una versión posterior. El operador Trident lanzado con 21.01.0 contiene un problema conocido que se ha solucionado en 21.01.1. Para más detalles, consulta el "detalles del problema en GitHub".

Paso 2: determina el método de instalación original

Para determinar qué versión usaste para instalar Trident originalmente:

1. Usa `kubectl get pods -n trident` para examinar los pods.
   - Si no hay ningún pod de operador, Trident se instaló usando `tridentctl`.
   - Si hay un pod de operador, Trident se instaló usando el operador Trident, ya sea manualmente o usando Helm.
2. Si hay un pod de operador, usa `kubectl describe torc` para determinar si Trident se instaló usando Helm.
   - Si hay una etiqueta Helm, Trident se instaló usando Helm.
   - Si no hay etiqueta Helm, Trident se instaló manualmente usando el operador Trident.

Paso 3: selecciona un método de actualización

Generalmente, deberías actualizar usando el mismo método que usaste para la instalación inicial, aunque puedes "moverse entre métodos de instalación". Hay dos opciones para actualizar Trident.

- "Actualiza usando el operador Trident"

Te sugerimos que revises "Entiende el flujo de trabajo de actualización del operador" antes de actualizar con el operador.

*

Actualiza con el operador

Entiende el flujo de trabajo de actualización del operador

Antes de usar el operador Trident para actualizar Trident, debes entender los procesos en segundo plano que ocurren durante la actualización. Esto incluye cambios en el controlador Trident, el Pod de controlador y los Pods de nodo, y el nodo DaemonSet que permiten actualizaciones continuas.

Gestión de la actualización del operador Trident

Uno de los muchos "ventajas de usar el operador Trident" para instalar y actualizar Trident es el manejo automático de objetos de Trident y Kubernetes sin interrumpir los volúmenes montados existentes. De esta manera, Trident puede soportar actualizaciones con cero tiempo de inactividad o "actualizaciones continuas". En particular, el operador de Trident se comunica con el clúster de Kubernetes para:

- Elimina y vuelve a crear el deployment del controlador Trident y el DaemonSet de nodo.
- Reemplaza el Trident Controller Pod y los Trident Node Pods por nuevas versiones.

- Si un nodo no se actualiza, no impide que los nodos restantes se actualicen.
- Solo los nodos con un Trident Node Pod en ejecución pueden montar volúmenes.

Para más información sobre la arquitectura de Trident en el clúster de Kubernetes, consulta "Arquitectura de Trident".

Flujo de trabajo de actualización de Operator

Cuando inicias una actualización usando el operador Trident:

1. El **Trident operator**:
   a. Detecta la versión de Trident que tienes instalada actualmente (versión *n*).
   b. Actualiza todos los objetos de Kubernetes, incluidos CRDs, RBAC y Trident SVC.
   c. Elimina la implementación del controlador Trident para la versión *n*.
   d. Crea el despliegue del controlador Trident para la versión *n+1*.
2. **Kubernetes** crea Trident Controller Pod para *n+1*.
3. El **Trident operator**:
   a. Elimina el DaemonSet de nodo de Trident para *n*. El operador no espera a que termine el Pod de nodo.
   b. Crea el Daemonset del nodo Trident para *n+1*.
4. **Kubernetes** crea Trident Node Pods en nodos que no están ejecutando Trident Node Pod *n*. Esto garantiza que nunca haya más de un Trident Node Pod, de cualquier versión, en un nodo.

Actualiza una instalación de Trident usando el operador Trident o Helm

Puedes actualizar Trident usando el operador Trident, ya sea manualmente o usando Helm. Puedes actualizar desde una instalación del operador Trident a otra instalación del operador Trident, o actualizar desde una instalación de `tridentctl` a una versión del operador Trident. Revisa "Selecciona un método de actualización" antes de actualizar una instalación del operador Trident.

Actualizar una instalación manual

Puedes actualizar desde una instalación de operador Trident con alcance de clúster a otra instalación de operador Trident con alcance de clúster. Todas las versiones de Trident usan un operador con alcance de clúster.

Para actualizar desde Trident que se instaló utilizando el operador de espacio de nombres (versiones 20.07 a 20.10), usa las instrucciones de actualización para "tu versión instalada" de Trident.

Acerca de esta tarea

Trident proporciona un archivo de paquete que puedes usar para instalar el operador y crear objetos asociados para tu versión de Kubernetes.

- Para los clústeres que ejecutan Kubernetes 1.24, usa "bundle_pre_1_25.yaml".
- Para clústeres que ejecutan Kubernetes 1.25 o posterior, usa "bundle_post_1_25.yaml".

Antes de empezar

Asegúrate de estar usando un clúster de Kubernetes que esté ejecutando "una versión compatible de Kubernetes".

Pasos

1. Verifica tu versión de Trident:

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

2. Actualiza el `operator.yaml`, `tridentorchestrator_cr.yaml` y `post_1_25_bundle.yaml` con el registro y las rutas de imagen para la versión a la que estás actualizando (por ejemplo, 25.06), y el secreto correcto.
3. Elimina el operador Trident que se usó para instalar la instancia actual de Trident. Por ejemplo, si estás actualizando desde 25.02, ejecuta el siguiente comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Si personalizaste tu instalación inicial usando `TridentOrchestrator` atributos, puedes editar el objeto `TridentOrchestrator` para modificar los parámetros de instalación. Esto puede incluir cambios para especificar registros de imágenes de Trident y CSI reflejados para el modo sin conexión, habilitar registros de depuración o especificar secretos de extracción de imágenes.
5. Instala Trident usando el archivo YAML del paquete correcto para tu entorno, donde *<bundle.yaml>* es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes. Por ejemplo, si estás instalando Trident 25.06.0, ejecuta el siguiente comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Edita el trident torc para incluir la imagen 25.06.0.

Actualizar una instalación de Helm

Puedes actualizar una instalación de Trident Helm.

> Al actualizar un clúster de Kubernetes de 1.24 a 1.25 o posterior que tenga Trident instalado, debes actualizar values.yaml para establecer `excludePodSecurityPolicy` en `true` o agregar `--set excludePodSecurityPolicy=true` al comando `helm upgrade` antes de poder actualizar el clúster.

Si ya actualizaste tu clúster de Kubernetes de la versión 1.24 a la 1.25 sin actualizar el helm de Trident, la actualización de helm fallará. Para que la actualización de helm funcione, realiza estos pasos como requisitos previos:

1. Instala el complemento helm-mapkubeapis desde https://github.com/helm/helm-mapkubeapis.
2. Haz una prueba en seco de la versión de Trident en el espacio de nombres donde está instalado. Esto

muestra los recursos que se limpiarán.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Realiza una ejecución completa con helm para hacer la limpieza.

```
helm mapkubeapis trident  --namespace trident
```

Pasos

1. Si "instalé Trident usando Helm", puedes usar `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0` para actualizar en un solo paso. Si no agregaste el repositorio de Helm o no puedes usarlo para actualizar:
   a. Descarga la última versión de Trident desde "la sección *Assets* en GitHub".
   b. Usa el `helm upgrade` comando donde `trident-operator-25.10.0.tgz` refleja la versión a la que quieres actualizar.

```
helm upgrade <name> trident-operator-25.10.0.tgz
```

> Si configuras opciones personalizadas durante la instalación inicial (como especificar registros privados o reflejados para las imágenes de Trident y CSI), agrega el `helm upgrade` comando usando `--set` para asegurarte de que esas opciones estén incluidas en el comando de actualización, de lo contrario los valores se restablecerán a los predeterminados.

2. Ejecuta `helm list` para verificar que tanto el chart como la versión de la app se hayan actualizado. Ejecuta `tridentctl logs` para revisar cualquier mensaje de depuración.

Actualiza desde una `tridentctl` instalación a Trident operator

Puedes actualizar a la última versión del operador Trident desde una `tridentctl` instalación. Los backends y PVC existentes estarán disponibles automáticamente.

> Antes de cambiar entre métodos de instalación, revisa "Cambiar entre métodos de instalación".

Pasos

1. Descarga la última versión de Trident.

```
# Download the release required [25.10.0]
mkdir 25.10.0
cd 25.10.0
wget
https://github.com/NetApp/trident/releases/download/v25.10.0/trident-
installer-25.10.0.tar.gz
tar -xf trident-installer-25.10.0.tar.gz
cd trident-installer
```

2. Crea el `tridentorchestrator` CRD a partir del manifiesto.

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

3. Implementa el operador con ámbito de clúster en el mismo espacio de nombres.

```
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. Crea un `TridentOrchestrator`CR para instalar 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. Confirma que Trident se actualizó a la versión prevista.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.10.0
```

Actualiza con tridentctl

Puedes actualizar fácilmente una instalación existente de Trident usando `tridentctl`.

Acerca de esta tarea

Desinstalar y volver a instalar Trident actúa como una actualización. Cuando desinstalas Trident, la Persistent Volume Claim (PVC) y el Persistent Volume (PV) usados por el deployment de Trident no se eliminan. Los PV que ya se hayan aprovisionado seguirán disponibles mientras Trident esté fuera de línea, y Trident aprovisionará volúmenes para cualquier PVC que se cree en ese tiempo después de que vuelva a estar en línea.

Antes de empezar

Revisa "Selecciona un método de actualización" antes de actualizar usando `tridentctl`.

Pasos

1. Ejecuta el comando de desinstalación en `tridentctl` para eliminar todos los recursos asociados con Trident excepto los CRDs y los objetos relacionados.

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

2. Vuelve a instalar Trident. Consulta "Instala Trident usando tridentctl".

> No interrumpas el proceso de actualización. Asegúrate de que el instalador se ejecute hasta el final.

Administra Trident usando tridentctl

El "Paquete de instalación de Trident" incluye la `tridentctl` utilidad de línea de comandos para proporcionar un acceso sencillo a Trident. Los usuarios de Kubernetes con privilegios suficientes pueden usarla para instalar Trident o gestionar el namespace que contiene el pod de Trident.

Comandos y banderas globales

Puedes ejecutar `tridentctl help` para obtener una lista de comandos disponibles para `tridentctl` o añadir la bandera `--help` a cualquier comando para ver una lista de opciones y banderas para ese comando específico.

```
tridentctl [command] [--optional-flag]
```

La utilidad Trident `tridentctl` admite los siguientes comandos y banderas globales.

Comandos

`create`
Agrega un recurso a Trident.

`delete`
Elimina uno o varios recursos de Trident.

`get`
Obtén uno o más recursos de Trident.

`help`
Ayuda sobre cualquier comando.

`images`
Imprime una tabla de las imágenes de contenedores que necesita Trident.

`import`
Importa un recurso existente a Trident.

`install`
Instala Trident.

`logs`
Imprime los registros de Trident.

`send`
Envía un recurso desde Trident.

`uninstall`
Desinstala Trident.

`update`
Modifica un recurso en Trident.

`update backend state`
Suspende temporalmente las operaciones de backend.

`upgrade`
Actualiza un recurso en Trident.

`version`
Imprime la versión de Trident.

Banderas globales

`-d, --debug`
Salida de depuración.

`-h, --help`
Ayuda para `tridentctl`.

`-k, --kubeconfig string`
Especifica la `KUBECONFIG` ruta para ejecutar comandos localmente o de un clúster de Kubernetes a otro.

> Como alternativa, puedes exportar la `KUBECONFIG` variable para que apunte a un clúster de Kubernetes específico y emitir `tridentctl` comandos a ese clúster.

`-n, --namespace string`
Espacio de nombres de la implementación de Trident.

`-o, --output string`
Formato de salida. Uno de json|yaml|name|wide|ps (predeterminado).

`-s, --server string`
Dirección/puerto de la interfaz de REST de Trident.

> La interfaz REST de Trident se puede configurar para escuchar y servir en 127.0.0.1 (para IPv4) o [::1] (para IPv6) solamente.

Opciones de comando y flags

crear

Usa el `create` comando para agregar un recurso a Trident.

```
tridentctl create [option]
```

Opciones

`backend`: agrega un backend a Trident.

borrar

Usa el comando `delete` para eliminar uno o más recursos de Trident.

```
tridentctl delete [option]
```

Opciones

`backend`: eliminar uno o más backends de almacenamiento de Trident.
`snapshot`: eliminar una o más instantáneas de volumen de Trident.
`storageclass`: eliminar una o más clases de almacenamiento de Trident.

`volume`: eliminar uno o más volúmenes de almacenamiento de Trident.

obtener

Usa el `get` comando para obtener uno o más recursos de Trident.

```
tridentctl get [option]
```

Opciones

`backend`: Obtén uno o más backends de almacenamiento de Trident.
`snapshot`: Obtén una o más instantáneas de Trident.
`storageclass`: Obtén una o más clases de almacenamiento de Trident.
`volume`: Obtén uno o más volúmenes de Trident.

Banderas

`-h`, `--help`: Ayuda para volúmenes.
`--parentOfSubordinate string`: Limitar consulta a volumen de origen subordinado.
`--subordinateOf string`: Limitar consulta a subordinados de volumen.

imágenes

Usa `images`banderas para imprimir una tabla de las imágenes de contenedores que necesita Trident.

```
tridentctl images [flags]
```

Banderas

`-h`, `--help`: ayuda para imágenes.
`-v`, `--k8s-version string`: versión semántica del clúster de Kubernetes.

importar volumen

Usa el `import volume` comando para importar un volumen existente a Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Alias

`volume`, `v`

Banderas

`-f`, `--filename string`: ruta al archivo YAML o JSON PVC.
`-h`, `--help`: ayuda para el volumen.
`--no-manage`: crear solo PV/PVC. No asumas la gestión del ciclo de vida del volumen.

instalar

Usa las `install` banderas para instalar Trident.

```
tridentctl install [flags]
```

Banderas

`--autosupport-image string`: La imagen del contenedor para Autosupport Telemetry

(predeterminado "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: La dirección/puerto de un proxy para enviar Autosupport Telemetry.
`--enable-node-prep`: Intentar instalar los paquetes requeridos en los nodos.
`--generate-custom-yaml`: Generar archivos YAML sin instalar nada.
`-h, --help`: Ayuda para instalar.
`--http-request-timeout`: Anular el tiempo de espera de la solicitud HTTP para la API de REST del controlador Trident (predeterminado 1m30s).
`--image-registry string`: La dirección/puerto de un registro interno de imágenes.
`--k8s-timeout duration`: El tiempo de espera para todas las operaciones de Kubernetes (predeterminado 3m0s).
`--kubelet-dir string`: La ubicación en el host del estado interno de kubelet (predeterminado "/var/lib/kubelet").
`--log-format string`: El formato de registro de Trident (text, json) (predeterminado "text").
`--node-prep`: Permite que Trident prepare los nodos del clúster de Kubernetes para gestionar volúmenes usando el protocolo de almacenamiento de datos especificado. **Actualmente, `iscsi` es el único valor admitido. Beginning with OpenShift 4.19, la versión mínima de Trident compatible con esta función es 25.06.1.**
`--pv string`: El nombre del PV heredado que usa Trident, asegúrate de que no exista (predeterminado "trident").
`--pvc string`: El nombre del PVC heredado que usa Trident, asegúrate de que no exista (predeterminado "trident").
`--silence-autosupport`: No enviar paquetes de autosupport a NetApp automáticamente (predeterminado true).
`--silent`: Deshabilitar la mayoría de la salida durante la instalación.
`--trident-image string`: La imagen de Trident que se va a instalar.
`--k8s-api-qps`: El límite de consultas por segundo (QPS) para las solicitudes de la API de Kubernetes (predeterminado 100; opcional).
`--use-custom-yaml`: Usar cualquier archivo YAML existente que haya en el directorio de setup.
`--use-ipv6`: Usar IPv6 para la comunicación de Trident.

registros

Usa `logs`banderas para imprimir los registros de Trident.

```
tridentctl logs [flags]
```

Banderas

`-a, --archive`: Crea un archivo de soporte con todos los registros a menos que se especifique lo contrario.
`-h, --help`: Ayuda para registros.
`-l, --log string`: Registro de Trident para mostrar. Uno de trident|auto|trident-operator|all (predeterminado "auto").
`--node string`: El nombre de nodo de Kubernetes del que se recopilarán los registros del pod del nodo.
`-p, --previous`: Obtén los registros de la instancia anterior del contenedor si existe.
`--sidecars`: Obtén los registros de los contenedores sidecar.

enviar

Usa el `send` comando para enviar un recurso desde Trident.

```
tridentctl send [option]
```

Opciones

`autosupport`: Envía un archivo de Autosupport a NetApp.

desinstalar

Usa `uninstall` flags para desinstalar Trident.

```
tridentctl uninstall [flags]
```

Banderas

`-h, --help`: ayuda para la desinstalación.
`--silent`: deshabilitar la mayoría de las salidas durante la desinstalación.

actualizar

Usa el `update` comando para modificar un recurso en Trident.

```
tridentctl update [option]
```

Opciones

`backend`: actualiza un backend en Trident.

actualiza el estado del backend

Usa el `update backend state` comando para suspender o reanudar las operaciones del backend.

```
tridentctl update backend state <backend-name> [flag]
```

Puntos a considerar

- Si se crea un backend utilizando un TridentBackendConfig (tbc), el backend no se puede actualizar usando un archivo `backend.json`.
- Si el `userState` se ha establecido en un tbc, no se puede modificar usando el comando `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Para recuperar la capacidad de configurar el `userState` mediante tridentctl después de que se haya configurado mediante tbc, el campo `userState` debe eliminarse de tbc. Esto se puede hacer usando el comando `kubectl edit tbc`. Después de eliminar el campo `userState`, puedes usar el comando `tridentctl update backend state` para cambiar el `userState` de un backend.
- Usa el `tridentctl update backend state` para cambiar el `userState`. También puedes actualizar el `userState` usando `TridentBackendConfig` o `backend.json` archivo; esto activa una reinicialización completa del backend y puede llevar mucho tiempo.

  #### Banderas

  `-h, --help`: Ayuda para el estado del backend.
  `--user-state`: Configura en `suspended` para pausar las operaciones del backend. Configura en `normal` para reanudar las operaciones del backend. Cuando se configura en `suspended`:

- `AddVolume` y `Import Volume` están en pausa.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`,

`ReconcileNodeAccess` siguen estando disponibles.

También puedes actualizar el estado del backend usando `userState` el campo en el archivo de configuración del backend `TridentBackendConfig` o `backend.json`. Para más información, consulta "Opciones para gestionar backends" y "Realiza la gestión del backend con kubectl".

Ejemplo:

JSON

Sigue estos pasos para actualizar el `userState` usando el `backend.json` archivo:

1. Edita el `backend.json` archivo para incluir el `userState` campo con su valor establecido en 'suspended'.
2. Actualiza el backend usando el `tridentctl update backend` comando y la ruta al archivo `backend.json` actualizado.

   Ejemplo: `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

Puedes editar el tbc después de aplicarlo usando el `kubectl edit <tbc-name> -n <namespace>` comando. El siguiente ejemplo actualiza el estado del backend a suspendido usando la opción `userState: suspended`:

```
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
```

versión

Usa `version` banderas para imprimir la versión de `tridentctl` y el servicio Trident en ejecución.

```
tridentctl version [flags]
```

Banderas

`--client`: Solo versión de cliente (no requiere servidor).
`-h, --help`: Ayuda para la versión.

Compatibilidad con complementos

Tridentctl admite plugins similares a kubectl. Tridentctl detecta un plugin si el nombre del archivo binario del plugin sigue el esquema "tridentctl-<plugin>" y el binario está ubicado en una carpeta listada en la variable de entorno PATH. Todos los plugins detectados se muestran en la sección de plugins de la ayuda de tridentctl. Opcionalmente, también puedes limitar la búsqueda especificando una carpeta de plugins en la variable de entorno TRIDENTCTL_PLUGIN_PATH (ejemplo: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Si se usa la variable, tridentctl busca solo en la carpeta especificada.

Monitorea Trident

Trident proporciona un conjunto de puntos finales de métricas de Prometheus que puedes usar para monitorear el rendimiento de Trident.

Descripción general

Las métricas proporcionadas por Trident te permiten hacer lo siguiente:

- Monitorea el estado y la configuración de Trident. Puedes examinar qué tan exitosas son las operaciones y si puede comunicarse con los backends como se espera.
- Examina la información de uso del backend y entiende cuántos volúmenes se aprovisionan en un backend y la cantidad de espacio consumido, y así sucesivamente.
- Mantén un mapeo de la cantidad de volúmenes aprovisionados en los backends disponibles.
- Monitorea el rendimiento. Puedes ver cuánto tarda Trident en comunicarse con los backends y realizar operaciones.

> De forma predeterminada, las métricas de Trident se exponen en el puerto de destino `8001` en el endpoint `/metrics`. Estas métricas están **habilitadas por defecto** cuando instalas Trident. También puedes configurar el consumo de métricas de Trident a través de HTTPS en el puerto `8444`.

Lo que necesitas

- Un clúster de Kubernetes con Trident instalado.
- Una instancia de Prometheus. Esto puede ser un "Implementación de Prometheus en contenedores" o puedes elegir ejecutar Prometheus como un "aplicación nativa".

Paso 1: define un objetivo de Prometheus

Deberías definir un objetivo de Prometheus para recopilar las métricas y obtener información sobre los

backends que Trident administra, los volúmenes que crea y así sucesivamente. Consulta "Documentación de Prometheus Operator".

Paso 2: Crea un ServiceMonitor de Prometheus

Para consumir las métricas de Trident, debes crear un Prometheus ServiceMonitor que supervise el `trident-csi` servicio y escuche en el `metrics` puerto. Un ejemplo de ServiceMonitor se ve así:

```
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
```

Esta definición de ServiceMonitor recupera las métricas devueltas por el `trident-csi` servicio y busca específicamente el `metrics` endpoint del servicio. Como resultado, Prometheus ahora está configurado para entender las métricas de Trident.

Además de las métricas disponibles directamente desde Trident, kubelet expone muchas `kubelet_volume_*` métricas a través de su propio endpoint de métricas. Kubelet puede proporcionar información sobre los volúmenes que están conectados, los pods y otras operaciones internas que maneja. Consulta "aquí".

Consume métricas de Trident a través de HTTPS

Para consumir métricas de Trident a través de HTTPS (puerto 8444), debes modificar la definición de ServiceMonitor para incluir la configuración TLS. También necesitas copiar el `trident-csi` secreto desde el `trident` namespace al namespace donde se está ejecutando Prometheus. Puedes hacer esto usando el siguiente comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un ejemplo de ServiceMonitor para métricas HTTPS se ve así:

```
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 admite métricas HTTPS en todos los métodos de instalación: tridentctl, Helm chart y Operator:

- Si estás usando el `tridentctl install` comando, puedes pasar la `--https-metrics` bandera para habilitar las métricas HTTPS.
- Si estás usando el gráfico Helm, puedes configurar el `httpsMetrics` parámetro para habilitar las métricas HTTPS.
- Si estás usando archivos YAML, puedes agregar la `--https_metrics` bandera al `trident-main` contenedor en el `trident-deployment.yaml` archivo.

Paso 3: Consultar métricas de Trident con PromQL

PromQL es bueno para crear expresiones que devuelven series de tiempo o datos tabulares.

Aquí hay algunas consultas PromQL que puedes usar:

Obtén información de salud de Trident

- **Porcentaje de respuestas HTTP 2XX de Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Porcentaje de respuestas REST de Trident mediante código de estado**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duración media en ms de las operaciones realizadas por Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtén información sobre el uso de Trident

- **Tamaño promedio del volumen**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espacio total de volumen aprovisionado por cada backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtén el uso del volumen individual

> Esto solo está habilitado si también se recopilan métricas de kubelet.

- **Porcentaje de espacio utilizado para cada volumen**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Conoce la telemetría de Trident AutoSupport

De forma predeterminada, Trident envía métricas de Prometheus e información básica del backend a NetApp cada día.

- Para evitar que Trident envíe métricas de Prometheus e información básica del backend a NetApp, pasa la `--silence-autosupport` bandera durante la instalación de Trident.
- Trident también puede enviar registros de contenedores a NetApp Support bajo demanda mediante `tridentctl send autosupport`. Necesitarás activar Trident para que suba sus registros. Antes de enviar los registros, deberías aceptar la NetApp "política de privacidad".
- A menos que se especifique, Trident recupera los registros de las últimas 24 horas.
- Puedes especificar el periodo de retención de registros con la `--since` bandera. Por ejemplo: `tridentctl send autosupport --since=1h`. Esta información se recopila y se envía mediante un `trident-autosupport` contenedor que se instala junto con Trident. Puedes obtener la imagen del contenedor en "Trident AutoSupport".
- Trident AutoSupport no recopila ni transmite información personal identificable (PII) ni información personal. Viene con una "CLUF" que no aplica a la propia imagen del contenedor de Trident. Puedes aprender más sobre el compromiso de NetApp con la seguridad y la confianza de los datos "aquí".

Un ejemplo de carga útil enviada por Trident se ve así:

```
---
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
```

- Los mensajes de AutoSupport se envían al endpoint de AutoSupport de NetApp. Si estás usando un registro privado para almacenar imágenes de contenedor, puedes usar la bandera `--image-registry`.
- También puedes configurar las URL de proxy generando los archivos YAML de instalación. Esto se puede hacer usando `tridentctl install --generate-custom-yaml` para crear los archivos YAML y añadiendo el argumento `--proxy-url` para el contenedor `trident-autosupport` en `trident-`

`deployment.yaml`.

Desactivar métricas de Trident

Para **deshabilitar** que se informen las métricas, debes generar YAML personalizados (usando la `--generate-custom-yaml` bandera) y editarlos para quitar la `--metrics` bandera de ser invocada para el contenedor `trident-main`.

Desinstala Trident

Debes usar el mismo método para desinstalar Trident que usaste para instalar Trident.

Acerca de esta tarea

- Si necesitas una solución para errores observados después de una actualización, problemas de dependencias o una actualización fallida o incompleta, deberías desinstalar Trident y reinstalar la versión anterior usando las instrucciones específicas para ese "versión". Esta es la única forma recomendada de *degradar* a una versión anterior.
- Para facilitar la actualización y reinstalación, desinstalar Trident no elimina los CRD ni los objetos relacionados creados por Trident. Si necesitas eliminar Trident por completo y todos sus datos, consulta "Eliminar por completo Trident y CRDs".

Antes de empezar

Si vas a decomisionar clústeres de Kubernetes, debes eliminar todas las aplicaciones que usan volúmenes creados por Trident antes de desinstalar. Esto asegura que los PVC se despublican en los nodos de Kubernetes antes de eliminarlos.

Determina el método de instalación original

Deberías usar el mismo método para desinstalar Trident que usaste para instalarlo. Antes de desinstalar, verifica qué versión usaste para instalar Trident originalmente.

1. Usa `kubectl get pods -n trident` para examinar los pods.
   - Si no hay ningún pod de operador, Trident se instaló usando `tridentctl`.
   - Si hay un pod de operador, Trident se instaló usando el operador Trident, ya sea manualmente o usando Helm.
2. Si hay un pod de operador, usa `kubectl describe tproc trident` para determinar si Trident se instaló usando Helm.
   - Si hay una etiqueta Helm, Trident se instaló usando Helm.
   - Si no hay etiqueta Helm, Trident se instaló manualmente usando el operador Trident.

Desinstala una instalación del operador Trident

Puedes desinstalar una instalación de operador Trident manualmente o usando Helm.

Desinstala la instalación manual

Si instalaste Trident usando el operador, puedes desinstalarlo haciendo una de las siguientes cosas:

1. **Edita `TridentOrchestrator` CR y establece la bandera de desinstalación**:

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

Cuando el `uninstall` flag se establece en `true`, el operador de Trident desinstala Trident, pero no elimina el TridentOrchestrator en sí. Deberías limpiar el TridentOrchestrator y crear uno nuevo si quieres instalar Trident otra vez.

2. **Eliminar `TridentOrchestrator`**: al eliminar el `TridentOrchestrator` CR que se utilizó para desplegar Trident, le indicas al operador que desinstale Trident. El operador procesa la eliminación de `TridentOrchestrator` y procede a eliminar el despliegue de Trident y el daemonset, borrando los pods de Trident que había creado como parte de la instalación.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Desinstalar la instalación de Helm

Si instalaste Trident usando Helm, puedes desinstalarlo usando `helm uninstall`.

```
#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
```

Desinstala una `tridentctl` instalación

Usa el comando `uninstall` en `tridentctl` para eliminar todos los recursos asociados con Trident excepto los CRDs y los objetos relacionados:

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

Trident para Docker

Requisitos previos para el despliegue

Tienes que instalar y configurar los requisitos previos de protocolo necesarios en tu host antes de poder desplegar Trident.

Verifica los requisitos

- Verifica que tu implementación cumple con todos los "requisitos".
- Verifica que tienes instalada una versión compatible de Docker. Si tu versión de Docker está desactualizada, "instálalo o actualízalo".

```
docker --version
```

- Verifica que los requisitos previos del protocolo estén instalados y configurados en tu host.

Herramientas NFS

Instala las herramientas NFS usando los comandos para tu sistema operativo.

RHEL 8+

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

Ubuntu

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

Reinicia tus nodos trabajadores después de instalar las herramientas NFS para evitar fallos al adjuntar volúmenes a los contenedores.

herramientas iSCSI

Instala las herramientas iSCSI usando los comandos para tu sistema operativo.

RHEL 8+

1. Instala los siguientes paquetes del sistema:

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

2. Verifica que la versión de iscsi-initiator-utils sea 6.2.0.874-2.el7 o posterior:

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

3. Configura el escaneo en manual:

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

4. Activa la multivía:

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

> Asegúrate de que `etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `iscsid` y `multipathd` están en funcionamiento:

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

6. Activa e inicia `iscsi`:

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

Ubuntu

1. Instala los siguientes paquetes del sistema:

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

2. Verifica que la versión de open-iscsi sea 2.0.874-5ubuntu2.10 o posterior (para bionic) o 2.0.874-7.1ubuntu6.1 o posterior (para focal):

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

3. Configura el escaneo en manual:

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

4. Activa la multivía:

```
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
```

> Asegúrate de que `etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

5. Asegúrate de que `open-iscsi` y `multipath-tools` están habilitados y en funcionamiento:

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

Herramientas NVMe

Instala las herramientas NVMe usando los comandos para tu sistema operativo.

- NVMe requiere RHEL 9 o posterior.
- Si la versión del kernel de tu nodo de Kubernetes es demasiado antigua o si el paquete NVMe no está disponible para tu versión del kernel, puede que tengas que actualizar la versión del kernel de tu nodo a una que tenga el paquete NVMe.

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
```

Herramientas FC

Instala las herramientas de FC usando los comandos para tu sistema operativo.

- Cuando uses nodos trabajadores que ejecutan RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) con FC PVs, especifica el `discard` mountOption en la StorageClass para realizar la reclamación de espacio en línea. Consulta "Documentación de Red Hat".

RHEL 8+

1. Instala los siguientes paquetes del sistema:

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

2. Activa la multivía:

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

> Asegúrate de que `etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipathd` está funcionando:

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

Ubuntu

1. Instala los siguientes paquetes del sistema:

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

2. Activa la multivía:

```
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
```

> Asegúrate de que `etc/multipath.conf` contiene `find_multipaths no` bajo `defaults`.

3. Asegúrate de que `multipath-tools` está activado y en funcionamiento:

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

Implementa Trident

Trident para Docker proporciona integración directa con el ecosistema de Docker para plataformas de almacenamiento NetApp. Admite el aprovisionamiento y la gestión de recursos de almacenamiento desde la plataforma de almacenamiento a los hosts de Docker, con un marco para añadir plataformas adicionales en el futuro.

Múltiples instancias de Trident pueden ejecutarse simultáneamente en el mismo host. Esto permite conexiones simultáneas a varios sistemas de almacenamiento y tipos de almacenamiento, con la posibilidad de personalizar el almacenamiento usado para los volúmenes de Docker.

Lo que necesitas

Consulta "requisitos previos para el despliegue". Después de asegurarte de que se cumplen los requisitos previos, ya puedes desplegar Trident.

Método de plugin gestionado por Docker (versión 1.13/17.03 y posteriores)

Antes de empezar

Si has utilizado Trident antes de Docker 1.13/17.03 con el método daemon tradicional, asegúrate de detener el proceso Trident y reiniciar tu daemon Docker antes de usar el método de plugin gestionado.

1. Detén todas las instancias en ejecución:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Reinicia Docker.

```
systemctl restart docker
```

3. Asegúrate de tener instalado Docker Engine 17.03 (new 1.13) o posterior.

```
docker --version
```

Si tu versión no está actualizada "instala o actualiza tu instalación".

Pasos

1. Crea un archivo de configuración y especifica las opciones como se indica a continuación:
   - `config`: el nombre de archivo por defecto es `config.json`, pero puedes usar cualquier nombre que elijas especificando la opción `config` con el nombre de archivo. El archivo de configuración debe estar ubicado en el directorio `/etc/netappdvp` en el sistema host.
   - `log-level`: especifica el nivel de registro (`debug`, `info`, `warn`, `error`, `fatal`). El valor por defecto es `info`.

- `debug`: especifica si el registro de depuración está activado. El valor predeterminado es false. Anula log-level si es true.

  i. Crea una ubicación para el archivo de configuración:

  ```
  sudo mkdir -p /etc/netappdvp
  ```

  ii. Crea el archivo de configuración:

  ```
  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. Inicia Trident usando el sistema de plugins gestionado. Reemplaza `<version>` con la versión del plugin (xxx.xx.x) que estás usando.

   ```
   docker plugin install --grant-all-permissions --alias netapp
   netapp/trident-plugin:<version> config=myConfigFile.json
   ```

3. Empieza a usar Trident para consumir almacenamiento del sistema configurado.

   a. Crea un volumen llamado "firstVolume":

   ```
   docker volume create -d netapp --name firstVolume
   ```

   b. Crea un volumen por defecto cuando el contenedor se inicia:

   ```
   docker run --rm -it --volume-driver netapp --volume
   secondVolume:/my_vol alpine ash
   ```

   c. Elimina el volumen "firstVolume":

```
docker volume rm firstVolume
```

Método tradicional (versión 1.12 o anterior)

Antes de empezar

1. Asegúrate de tener Docker versión 1.10 o posterior.

   ```
   docker --version
   ```

   Si tu versión está desfasada, actualiza tu instalación.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   O, "sigue las instrucciones para tu distribución".

2. Asegúrate de que NFS y/o iSCSI están configurados para tu sistema.

Pasos

1. Instala y configura el complemento de volumen Docker de NetApp:
   a. Descarga y descomprime la aplicación:

      ```
      wget
      https://github.com/NetApp/trident/releases/download/10.0/trident-
      installer-25.10.0.tar.gz
      tar zxf trident-installer-25.10.0.tar.gz
      ```

   b. Ve a una ubicación en la ruta del bin:

      ```
      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. Crea una ubicación para el archivo de configuración:

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Crea el archivo de configuración:

      ```
      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. Después de colocar el binario y crear el archivo de configuración, inicia el demonio Trident usando el archivo de configuración que quieras.

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

> A menos que se especifique, el nombre predeterminado del controlador de volumen es "netapp".

Después de que el daemon esté iniciado, puedes crear y gestionar volúmenes usando la interfaz CLI de Docker.

3. Crea un volumen:

```
docker volume create -d netapp --name trident_1
```

4. Aprovisiona un volumen Docker al iniciar un contenedor:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Eliminar un volumen de Docker:

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

Inicia Trident al arrancar el sistema

Puedes encontrar un archivo de unidad de ejemplo para sistemas basados en systemd en `contrib/trident.service.example` en el repositorio Git. Para usar el archivo con RHEL, haz lo siguiente:

1. Copia el archivo en la ubicación correcta.

   Deberías usar nombres únicos para los archivos de unidad si tienes más de una instancia en ejecución.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Edita el archivo, cambia la descripción (línea 2) para que coincida con el nombre del controlador y la ruta del archivo de configuración (línea 9) para reflejar tu entorno.
3. Recarga systemd para que ingiera los cambios:

   ```
   systemctl daemon-reload
   ```

4. Habilita el servicio.

   Este nombre varía dependiendo de cómo nombraste el archivo en el directorio `/usr/lib/systemd/system`.

   ```
   systemctl enable trident
   ```

5. Inicia el servicio.

   ```
   systemctl start trident
   ```

6. Ver el estado.

   ```
   systemctl status trident
   ```

> Cada vez que modifiques el archivo de la unidad, ejecuta el comando `systemctl daemon-reload` para que esté al tanto de los cambios.

Actualizar o desinstalar Trident

Puedes actualizar Trident para Docker de forma segura sin ningún impacto en los volúmenes que están en uso. Durante el proceso de actualización habrá un breve período en el que los comandos `docker volume` dirigidos al plugin no tendrán éxito y

las aplicaciones no podrán montar volúmenes hasta que el plugin esté funcionando de nuevo. En la mayoría de las circunstancias, esto es cuestión de segundos.

Actualizar

Sigue los pasos a continuación para actualizar Trident para Docker.

Pasos

1. Lista los volúmenes existentes:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Deshabilita el complemento:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Actualiza el complemento:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> La versión 18.01 de Trident reemplaza al nDVP. Deberías actualizar directamente de la `netapp/ndvp-plugin` imagen a la `netapp/trident-plugin` imagen.

4. Habilita el complemento:

```
docker plugin enable netapp:latest
```

5. Verifica que el complemento esté habilitado:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verifica que los volúmenes sean visibles:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

> Si actualizas desde una versión antigua de Trident (pre-20.10) a Trident 20.10 o posterior, podrías encontrarte con un error. Para más información, consulta "Problemas conocidos". Si te encuentras con el error, primero deberías deshabilitar el complemento, luego eliminarlo y después instalar la versión de Trident requerida pasando un parámetro de configuración extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Desinstalar

Sigue los pasos de abajo para desinstalar Trident para Docker.

Pasos

1. Elimina cualquier volumen que haya creado el complemento.
2. Deshabilita el complemento:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Elimina el complemento:

```
docker plugin rm netapp:latest
```

Trabaja con volúmenes

Puedes crear, clonar y eliminar volúmenes fácilmente usando los comandos estándar `docker volume` con el nombre del controlador Trident especificado cuando lo necesites.

Crear un volumen

- Crea un volumen con un controlador usando el nombre predeterminado:

```
docker volume create -d netapp --name firstVolume
```

- Crea un volumen con una instancia específica de Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Si no especificas ningún "opciones", se usan los valores predeterminados del controlador.

- Anula el tamaño de volumen predeterminado. Mira el siguiente ejemplo para crear un volumen de 20 GiB con un driver:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Los tamaños de volumen se expresan como cadenas que contienen un valor entero con unidades opcionales (ejemplo: 10G, 20GB, 3TiB). Si no se especifican unidades, el valor predeterminado es G. Las unidades de tamaño pueden expresarse como potencias de 2 (B, KiB, MiB, GiB, TiB) o potencias de 10 (B, KB, MB, GB, TB). Las unidades abreviadas utilizan potencias de 2 (G = GiB, T = TiB, …).

Eliminar un volumen

- Elimina el volumen como cualquier otro volumen de Docker:

```
docker volume rm firstVolume
```

Cuando usas el `solidfire-san` driver, el ejemplo anterior elimina y purga el volumen.

Sigue los pasos a continuación para actualizar Trident para Docker.

Clonar un volumen

Cuando usas los controladores de almacenamiento `ontap-nas`, `ontap-san` y `solidfire-san`, Trident puede clonar volúmenes. Cuando usas los controladores `ontap-nas-flexgroup` o `ontap-nas-economy`, la clonación no es compatible. Crear un volumen nuevo a partir de un volumen existente dará como resultado la creación de una nueva instantánea.

- Inspecciona el volumen para enumerar las instantáneas:

```
docker volume inspect <volume_name>
```

- Crear un nuevo volumen a partir de un volumen existente. Esto resultará en que se cree una nueva instantánea:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- Crear un nuevo volumen a partir de una instantánea existente en un volumen. Esto no creará una nueva instantánea:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Ejemplo

```
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
```

Accede a volúmenes creados externamente

Puedes acceder a dispositivos de bloque creados externamente (o a sus clones) mediante contenedores usando Trident **solo** si no tienen particiones y si su sistema de archivos es compatible con Trident (por ejemplo: un `ext4`-formateado `/dev/sdc1` no será accesible mediante Trident).

Opciones de volumen específicas del controlador

Cada controlador de almacenamiento tiene un conjunto diferente de opciones, que puedes especificar al crear el volumen para personalizar el resultado. Mira abajo las opciones que aplican a tu sistema de almacenamiento configurado.

Usar estas opciones durante la creación del volumen es sencillo. Proporciona la opción y el valor usando el `-o` operador durante la operación en la CLI. Estos valores anulan cualquier valor equivalente del archivo de configuración JSON.

Opciones de volumen de ONTAP

Las opciones de creación de volumen para NFS, iSCSI y FC incluyen lo siguiente:

Opción	Descripción
`size`	El tamaño del volumen, por defecto es 1 GiB.
`spaceReserve`	Aprovisiona el volumen de forma fina o gruesa, el valor predeterminado es fino. Los valores válidos son `none` (aprovisionamiento fino) y `volume` (aprovisionamiento grueso).
`snapshotPolicy`	Esto establecerá la política de instantáneas en el valor deseado. El valor predeterminado es `none`, lo que significa que no se crearán instantáneas automáticamente para el volumen. A menos que lo modifique tu administrador de almacenamiento, existe una política llamada "default" en todos los sistemas ONTAP que crea y conserva seis instantáneas cada hora, dos diarias y dos semanales. Los datos conservados en una instantánea se pueden recuperar navegando al directorio `.snapshot` en cualquier directorio del volumen.

Opción	Descripción
`snapshotReserve`	Esto establecerá la reserva de instantáneas en el porcentaje deseado. El valor predeterminado es ningún valor, lo que significa que ONTAP seleccionará el snapshotReserve (normalmente 5%) si has seleccionado un snapshotPolicy, o 0% si el snapshotPolicy es ninguno. Puedes establecer el valor predeterminado de snapshotReserve en el archivo de configuración para todos los backends de ONTAP, y puedes usarlo como opción de creación de volúmenes para todos los backends de ONTAP excepto ontap-nas-economy.
`splitOnClone`	Al clonar un volumen, esto hará que ONTAP separe inmediatamente el clon de su volumen principal. El valor predeterminado es `false`. Algunos casos de uso para clonar volúmenes se benefician más si se separa el clon de su volumen principal justo al crearlo, porque es poco probable que haya oportunidades para eficiencias de almacenamiento. Por ejemplo, clonar una base de datos vacía puede ahorrar mucho tiempo pero poco almacenamiento, así que lo mejor es separar el clon de inmediato.
`encryption`	Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE".
`tieringPolicy`	Establece la política de niveles que se usará para el volumen. Esto decide si los datos se mueven al nivel de nube cuando se vuelven inactivos (fríos).

Las siguientes opciones adicionales son **solo** para NFS:

Opción	Descripción
`unixPermissions`	Esto controla el conjunto de permisos para el volumen en sí. Por defecto, los permisos se establecerán en `---rwxr-xr-x, o en notación numérica 0755, y `root` será el propietario. Cualquiera de los formatos, texto o numérico, funcionará.

Opción	Descripción
`snapshotDir`	Configurar esto en `true` hará que el directorio `.snapshot` sea visible para los clientes que acceden al volumen. El valor predeterminado es `false`, lo que significa que la visibilidad del directorio `.snapshot` está deshabilitada por defecto. Algunas imágenes, por ejemplo la imagen oficial de MySQL, no funcionan como se espera cuando el directorio `.snapshot` es visible.
`exportPolicy`	Establece la política de exportación que se utilizará para el volumen. El valor predeterminado es `default`.
`securityStyle`	Establece el estilo de seguridad que se usará para acceder al volumen. El valor predeterminado es `unix`. Los valores válidos son `unix` y `mixed`.

Las siguientes opciones adicionales son para iSCSI **solamente**:

Opción	Descripción
`fileSystemType`	Establece el sistema de archivos utilizado para formatear volúmenes iSCSI. El valor predeterminado es `ext4`. Los valores válidos son `ext3`, `ext4` y `xfs`.
`spaceAllocation`	Configurar esto en `false` desactivará la función de asignación de espacio del LUN. El valor predeterminado es `true`, lo que significa que ONTAP notifica al host cuando el volumen se queda sin espacio y el LUN en el volumen no puede aceptar escrituras. Esta opción también permite que ONTAP recupere espacio automáticamente cuando tu host elimina datos.

Ejemplos

Mira los ejemplos a continuación:

- Crea un volumen de 10 GiB:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Crea un volumen de 100 GiB con instantáneas:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Crea un volumen que tenga el bit setUID habilitado:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

El tamaño mínimo del volumen es 20 MiB.

Si no se especifica la reserva de instantáneas y la política de instantáneas es `none`, Trident usa una reserva de instantáneas del 0%.

- Crea un volumen sin política de instantáneas ni reserva de instantáneas:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crea un volumen sin política de instantáneas y una reserva de instantáneas personalizada del 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crea un volumen con una política de instantáneas y una reserva de instantáneas personalizada del 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Crea un volumen con una política de instantáneas y acepta la reserva de instantáneas predeterminada de ONTAP (normalmente 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opciones de volumen del software Element

Las opciones de software de Element exponen el tamaño y las políticas de calidad de servicio (QoS) asociadas al volumen. Cuando se crea el volumen, la política de QoS asociada se especifica usando la `-o type=service_level` nomenclatura.

El primer paso para definir un nivel de servicio QoS con el controlador Element es crear al menos un tipo y especificar el IOPS mínimo, máximo y en ráfaga asociado con un nombre en el archivo de configuración.

Otras opciones para crear volúmenes con el software Element incluyen las siguientes:

Opción	Descripción
`size`	El tamaño del volumen, por defecto es 1 GiB o entrada de configuración … "defaults": {"size": "5G"}.
`blocksize`	Usa 512 o 4096, por defecto 512 o la entrada de configuración DefaultBlockSize.

Ejemplo

Consulta el siguiente ejemplo de archivo de configuración con definiciones de QoS:

```
{
  "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
      }
    }
  ]
}
```

En la configuración anterior, tenemos tres definiciones de política: Bronce, Plata y Oro. Estos nombres son arbitrarios.

- Crea un volumen Gold de 10 GiB:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Crea un volumen Bronze de 100 GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Recopilar registros

Puedes recopilar los registros para ayudarte con la resolución de problemas. El método que uses para recopilar los registros varía según cómo estés ejecutando el complemento Docker.

Recopila registros para solucionar problemas

Pasos

1. Si estás ejecutando Trident utilizando el método de plugin gestionado recomendado (es decir, usando `docker plugin` comandos), míralos así:

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

   El nivel de registro estándar debería permitirte diagnosticar la mayoría de los problemas. Si ves que no es suficiente, puedes activar el registro de depuración.

2. Para activar el registro de depuración, instala el complemento con el registro de depuración activado:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   O, activa el registro de depuración cuando el plugin ya esté instalado:

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. Si ejecutas el propio binario en el host, los registros están disponibles en el directorio `/var/log/netappdvp` del host. Para activar el registro de depuración, especifica `-debug` cuando ejecutes el complemento.

Consejos generales para la resolución de problemas

- El problema más común con el que se encuentran los nuevos usuarios es una configuración incorrecta que impide que el plugin se inicialice. Cuando esto pasa, probablemente verás un mensaje como este cuando intentes instalar o habilitar el plugin:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  Esto significa que el plugin no pudo iniciarse. Por suerte, el plugin se ha creado con una capacidad de registro muy completa que debería ayudarte a diagnosticar la mayoría de los problemas con los que probablemente te encuentres.

- Si hay problemas con montar un PV en un contenedor, asegúrate de que `rpcbind` está instalado y en ejecución. Usa el gestor de paquetes necesario para el sistema operativo anfitrión y revisa si `rpcbind` está en ejecución. Puedes comprobar el estado del servicio rpcbind ejecutando un `systemctl status rpcbind` o su equivalente.

Gestiona varias instancias de Trident

Se necesitan múltiples instancias de Trident cuando deseas tener varias configuraciones de almacenamiento disponibles al mismo tiempo. La clave para tener múltiples instancias es darles nombres diferentes usando la opción `--alias` con el plugin en contenedores, o la opción `--volume-driver` al instanciar Trident en el host.

Pasos para el plugin gestionado por Docker (versión 1.13/17.03 o posterior)

1. Lanza la primera instancia especificando un alias y un archivo de configuración.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Lanza la segunda instancia, especificando un alias y un archivo de configuración diferentes.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Crea volúmenes especificando el alias como nombre del controlador.

   Por ejemplo, para el volumen gold:

   ```
   docker volume create -d gold --name ntapGold
   ```

   Por ejemplo, para el volumen silver:

   ```
   docker volume create -d silver --name ntapSilver
   ```

Pasos para el método tradicional (versión 1.12 o anterior)

1. Lanza el complemento con una configuración NFS usando un ID de controlador personalizado:

   ```
   sudo trident --volume-driver=netapp-nas --config=/path/to/config
   -nfs.json
   ```

2. Lanza el plugin con una configuración iSCSI usando un driver ID personalizado:

   ```
   sudo trident --volume-driver=netapp-san --config=/path/to/config
   -iscsi.json
   ```

3. Provisiona volúmenes Docker para cada instancia de driver:

   Por ejemplo, para NFS:

   ```
   docker volume create -d netapp-nas --name my_nfs_vol
   ```

   Por ejemplo, para iSCSI:

   ```
   docker volume create -d netapp-san --name my_iscsi_vol
   ```

Opciones de configuración del almacenamiento

Consulta las opciones de configuración disponibles para tus configuraciones Trident.

Opciones de configuración global

Estas opciones de configuración se aplican a todas las configuraciones de Trident, independientemente de la plataforma de almacenamiento que estés usando.

Opción	Descripción	Ejemplo
`version`	Número de versión del archivo de configuración	`1`
`storageDriverName`	Nombre del controlador de almacenamiento	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	Prefijo opcional para los nombres de los volúmenes. Predeterminado: `netappdvp_`.	`staging_`
`limitVolumeSize`	Restricción opcional en los tamaños de los volúmenes. Predeterminado: "" (no se aplica)	`10g`

No utilices `storagePrefix` (incluido el predeterminado) para los backends de Element. Por defecto, el `solidfire-san` driver ignorará esta configuración y no usará un prefijo. NetApp recomienda usar un tenantID específico para el mapeo de volúmenes de Docker o usar el atributo data, que se llena con la versión de Docker, la info del driver y el nombre sin procesar de Docker en los casos donde se haya hecho algún cambio de nombre.

Las opciones por defecto están disponibles para evitar tener que especificarlas en cada volumen que crees. La opción `size` está disponible para todos los tipos de controlador. Consulta la sección de configuración de ONTAP para ver un ejemplo de cómo establecer el tamaño de volumen por defecto.

Opción	Descripción	Ejemplo
`size`	Tamaño predeterminado opcional para nuevos volúmenes. Predeterminado: `1G`	`10G`

Configuración de ONTAP

Además de los valores de configuración global anteriores, cuando usas ONTAP, están disponibles las siguientes opciones de nivel superior.

Opción	Descripción	Ejemplo
`managementLIF`	Dirección IP del LIF de gestión de ONTAP. Puedes especificar un nombre de dominio completo (FQDN).	`10.0.0.1`

Opción	Descripción	Ejemplo
`dataLIF`	Dirección IP del protocolo LIF. **Controladores NAS ONTAP**: NetApp recomienda especificar `dataLIF`. Si no se proporciona, Trident obtiene los dataLIFs del SVM. Puedes especificar un nombre de dominio completo (FQDN) para usar en las operaciones de montaje NFS, lo que te permite crear un DNS round-robin para balancear la carga entre varios dataLIFs. **ONTAP SAN drivers**: no especifiques para iSCSI o FC. Trident usa "Mapa LUN selectivo de ONTAP" para descubrir los LIF de iSCSI o FC necesarios para establecer una sesión multipath. Se genera una advertencia si `dataLIF` se define explícitamente.	`10.0.0.2`
`svm`	Máquina virtual de almacenamiento que se va a usar (necesaria, si la LIF de gestión es una LIF de clúster)	`svm_nfs`
`username`	Nombre de usuario para conectarte al dispositivo de almacenamiento	`vsadmin`
`password`	Contraseña para conectarte al dispositivo de almacenamiento	`secret`
`aggregate`	Agregado para el aprovisionamiento (opcional; si se establece, debe asignarse a la SVM). Para el `ontap-nas-flexgroup` driver, esta opción se ignora. Todos los agregados asignados a la SVM se utilizan para aprovisionar un volumen FlexGroup.	`aggr1`
`limitAggregateUsage`	Opcional, falla el aprovisionamiento si el uso es superior a este porcentaje	`75%`

Opción	Descripción	Ejemplo
`nfsMountOptions`	Control detallado de las opciones de montaje NFS; por defecto "-o nfsvers=3". **Disponible solo para los controladores `ontap-nas` y `ontap-nas-economy`.** "Consulta la información de configuración del host NFS aquí".	`-o nfsvers=4`
`igroupName`	Trident crea y gestiona por nodo `igroups` como `netappdvp`. Este valor no puede cambiarse ni omitirse. **Disponible solo para el `ontap-san` driver.**	`netappdvp`
`limitVolumeSize`	Tamaño máximo de volumen que se puede solicitar.	`300g`
`qtreesPerFlexvol`	Máximo de qtrees por FlexVol, debe estar en el rango [50, 300], el valor predeterminado es 200. **Para el `ontap-nas-economy` controlador, esta opción permite personalizar el número máximo de qtrees por FlexVol.**	`300`
`sanType`	**Sólo se admite para `ontap-san` driver.** Usa para seleccionar `iscsi` para iSCSI, `nvme` para NVMe/TCP o `fcp` para SCSI sobre Fibre Channel (FC).	`iscsi` si está en blanco
`limitVolumePoolSize`	**Solo es compatible con los controladores `ontap-san-economy` y `ontap-san-economy`.** Limita los tamaños de FlexVol en los controladores ONTAP ontap-nas-economy y ontap-SAN-economy.	`300g`

Hay opciones predeterminadas disponibles para evitar tener que especificarlas en cada volumen que crees:

Opción	Descripción	Ejemplo
`spaceReserve`	Modo de reserva de espacio; `none` (thin provisioned) o `volume` (thick)	`none`

Opción	Descripción	Ejemplo
`snapshotPolicy`	Política de SnapVault a usar, por defecto es `none`	`none`
`snapshotReserve`	Porcentaje de reserva de snapshot, el valor predeterminado es "" para aceptar el valor predeterminado de ONTAP	`10`
`splitOnClone`	Divide un clon de su padre al crearlo, por defecto a `false`	`false`
`encryption`	Habilita NetApp Volume Encryption (NVE) en el nuevo volumen; el valor predeterminado es `false`. NVE debe tener licencia y estar habilitado en el clúster para usar esta opción. Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado. Para más información, consulta: "Cómo funciona Trident con NVE y NAE".	verdadero
`unixPermissions`	Opción NAS para volúmenes NFS aprovisionados, predeterminado a `777`	`777`
`snapshotDir`	Opción NAS para acceso al `.snapshot`directorio.	"verdadero" para NFSv4 "falso" para NFSv3
`exportPolicy`	Opción NAS para la política de exportación NFS que se va a usar, por defecto es `default`	`default`
`securityStyle`	Opción NAS para acceso al volumen NFS aprovisionado. NFS admite `mixed` y `unix` estilos de seguridad. El valor predeterminado es `unix`.	`unix`
`fileSystemType`	Opción SAN para seleccionar el tipo de sistema de archivos, predeterminado a `ext4`	`xfs`
`tieringPolicy`	Política de tiering a usar, el valor predeterminado es `none`.	`none`
`skipRecoveryQueue`	Durante la eliminación de volumen, omite la cola de recuperación en el almacenamiento y elimina el volumen inmediatamente.	``

Opciones de escalado

Los `ontap-nas` y `ontap-san` controladores crean un ONTAP FlexVol para cada volumen de Docker. ONTAP admite hasta 1000 FlexVols por nodo del clúster, con un máximo de 12,000 volúmenes FlexVol por clúster. Si tus requisitos de volumen de Docker se ajustan a esa limitación, el controlador `ontap-nas` es la solución NAS preferida gracias a las funciones adicionales que ofrecen los FlexVols, como las instantáneas granulares por volumen de Docker y la clonación.

Si necesitas más volúmenes de Docker de los que pueden ser acomodados por los límites de FlexVol, elige el `ontap-nas-economy` o el `ontap-san-economy` driver.

El `ontap-nas-economy`driver crea volúmenes Docker como Qtrees de ONTAP dentro de un grupo de FlexVol volúmenes administrados automáticamente. Los Qtrees ofrecen un escalado mucho mayor, hasta 100,000 por nodo del clúster y 2,400,000 por clúster, a costa de algunas funciones. El `ontap-nas-economy`driver no admite instantáneas granulares de volúmenes Docker ni clonación.

> El `ontap-nas-economy` controlador no es compatible actualmente con Docker Swarm, porque Docker Swarm no organiza la creación de volúmenes en varios nodos.

El `ontap-san-economy` controlador crea volúmenes Docker como LUN de ONTAP dentro de un grupo compartido de FlexVol volúmenes administrados automáticamente. De esta manera, cada FlexVol no está restringido a solo un LUN y ofrece mejor escalabilidad para cargas de trabajo SAN. Dependiendo de la matriz de almacenamiento, ONTAP admite hasta 16384 LUN por clúster. Como los volúmenes son LUN por debajo, este controlador admite snapshots y clonación granular de volúmenes Docker.

Elige el `ontap-nas-flexgroup` driver para aumentar el paralelismo en un solo volumen que puede crecer hasta el rango de petabytes con miles de millones de archivos. Algunos casos de uso ideales para FlexGroups incluyen IA/ML/DL, big data y analítica, compilaciones de software, streaming, repositorios de archivos, y así sucesivamente. Trident usa todos los agregados asignados a una SVM al aprovisionar un volumen FlexGroup. El soporte de FlexGroup en Trident también tiene en cuenta las siguientes consideraciones:

- Requiere ONTAP versión 9.2 o superior.
- Al momento de escribir esto, FlexGroups solo admiten NFS v3.
- Se recomienda habilitar los identificadores NFSv3 de 64 bits para la SVM.
- El tamaño mínimo recomendado de miembro/volumen de FlexGroup es 100 GiB.
- No se admite la clonación para volúmenes FlexGroup.

Para obtener información sobre FlexGroups y las cargas de trabajo que son adecuadas para FlexGroups, consulta la "NetApp FlexGroup volume Guía de mejores prácticas e implementación".

Para obtener funciones avanzadas y a gran escala en el mismo entorno, puedes ejecutar varias instancias del complemento Docker Volume, una usando `ontap-nas` y otra usando `ontap-nas-economy`.

Rol personalizado de ONTAP para Trident

Puedes crear un rol de clúster de ONTAP con privilegios mínimos para que no tengas que usar el rol de admin de ONTAP para realizar operaciones en Trident. Cuando incluyes el nombre de usuario en una configuración de backend de Trident, Trident usa el rol de clúster de ONTAP que creaste para realizar las operaciones.

Consulta "Generador de roles personalizados de Trident" para más información sobre cómo crear roles personalizados de Trident.

Usando ONTAP CLI

1. Crea un nuevo rol usando el siguiente comando:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Crea un nombre de usuario para el usuario Trident:

```
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. Asigna el rol al usuario:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Realiza los siguientes pasos en ONTAP System Manager:

1. **Crea un rol personalizado**:
   a. Para crear un rol personalizado a nivel de cluster, selecciona **Cluster > Settings**.

      (O) Para crear un rol personalizado a nivel de SVM, selecciona **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecciona el icono de flecha (→) junto a **Users and Roles**.
   c. Selecciona **+Add** en **Roles**.
   d. Define las reglas para el rol y haz clic en **Guardar**.
2. **Asigna el rol al usuario Trident**: + Realiza los siguientes pasos en la página **Usuarios y roles**:
   a. Selecciona el icono Add **+** en **Usuarios**.
   b. Selecciona el nombre de usuario requerido y elige un rol en el menú desplegable de **Role**.
   c. Haz clic en **Guardar**.

Consulta las siguientes páginas para obtener más información:

- "Roles personalizados para la administración de ONTAP" o "Define roles personalizados"
- "Trabaja con roles y usuarios"

Ejemplo de archivos de configuración de ONTAP

Ejemplo de NFS para el controlador <code>ontap-nas</code>

```
{
  "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"
  }
}
```

Ejemplo de NFS para <code>ontap-nas-flexgroup</code> driver

```
{
  "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"
  }
}
```

Ejemplo de NFS para el controlador <code>ontap-nas-economy</code>

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
```

ejemplo de iSCSI para <code>ontap-san</code> driver

```
{
  "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"
}
```

Ejemplo de NFS para el controlador <code>ontap-san-economy</code>

```
{
  "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"
}
```

Ejemplo de NVMe/TCP para el controlador <code>ontap-san</code>

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

Ejemplo de SCSI sobre FC para <code>ontap-san</code> driver

```
{
  "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
}
```

Configuración del software Element

Además de los valores de configuración global, cuando usas el software Element (NetApp HCI/SolidFire), estas opciones están disponibles.

Opción	Descripción	Ejemplo
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	dirección IP y puerto iSCSI	10.0.0.7:3260
`TenantName`	SolidFireF Tenant que se va a usar (se crea si no se encuentra)	`docker`

Opción	Descripción	Ejemplo
`InitiatorIFace`	Especifica la interfaz cuando restrinjas el tráfico iSCSI a una interfaz que no sea la predeterminada	`default`
`Types`	Especificaciones de QoS	Mira el ejemplo de abajo
`LegacyNamePrefix`	Prefijo para las instalaciones actualizadas de Trident. Si usaste una versión de Trident anterior a la 1.3.2 y haces una actualización con volúmenes existentes, tendrás que establecer este valor para acceder a tus volúmenes antiguos que se asignaron mediante el método de nombre de volumen.	`netappdvp-`

El `solidfire-san` controlador no es compatible con Docker Swarm.

Ejemplo de archivo de configuración del software Element

```
{
  "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
      }
    }
  ]
}
```

Problemas conocidos y limitaciones

Encuentra información sobre problemas conocidos y limitaciones al usar Trident con Docker.

Al actualizar Trident Docker Volume Plugin a 20.10 y versiones posteriores desde versiones anteriores, la actualización falla con el error no such file or directory.

Solución alternativa

1. Desactiva el plugin.

```
docker plugin disable -f netapp:latest
```

2. Elimina el plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstala el plugin proporcionando el parámetro extra `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Los nombres de los volúmenes deben tener al menos 2 caracteres de longitud.

Esta es una limitación del cliente Docker. El cliente interpretará un nombre de un solo carácter como una ruta de Windows. "Consulta el bug 25773"

Docker Swarm tiene ciertos comportamientos que impiden que Trident lo admita con cada combinación de almacenamiento y controlador.

- Docker Swarm actualmente utiliza el nombre del volumen en lugar del ID del volumen como identificador único del volumen.
- Las solicitudes de volumen se envían simultáneamente a cada nodo en un clúster Swarm.
- Los plugins de volumen (incluido Trident) deben ejecutarse de forma independiente en cada nodo de un clúster Swarm. Debido al funcionamiento de ONTAP y a cómo los controladores `ontap-nas` y `ontap-san` funcionan, son los únicos que pueden operar dentro de estas limitaciones.

El resto de los controladores están sujetos a problemas como las condiciones de carrera que pueden dar lugar a la creación de un gran número de volúmenes para una sola solicitud sin un claro "ganador"; por ejemplo, Element tiene una función que permite que los volúmenes tengan el mismo nombre pero diferentes IDs.

NetApp ha proporcionado comentarios al equipo de Docker, pero no tiene ninguna indicación de futuras acciones.

Si se está aprovisionando un FlexGroup, ONTAP no aprovisiona un segundo FlexGroup si el segundo FlexGroup tiene uno o más agregados en común con el FlexGroup que se está aprovisionando.

Buenas prácticas y recomendaciones

Despliegue

Usa las recomendaciones que se indican aquí cuando despliegues Trident.

Implementa en un espacio de nombres dedicado

"Espacios de nombres" proporcionan separación administrativa entre diferentes aplicaciones y son una barrera para compartir recursos. Por ejemplo, una PVC de un espacio de nombres no puede consumirse desde otro. Trident proporciona recursos de PV a todos los espacios de nombres en el clúster de Kubernetes y, en consecuencia, aprovecha una cuenta de servicio que tiene privilegios elevados.

Además, el acceso al pod Trident podría permitir que un usuario acceda a las credenciales del sistema de almacenamiento y a otra información sensible. Es importante asegurarse de que los usuarios de las aplicaciones y las aplicaciones de gestión no tengan la capacidad de acceder a las definiciones de objetos Trident o a los propios pods.

Usa cuotas y límites de rango para controlar el consumo de almacenamiento

Kubernetes cuenta con dos funciones que, cuando se combinan, proporcionan un potente mecanismo para limitar el consumo de recursos por parte de las aplicaciones. El "mecanismo de cuota de almacenamiento" permite al administrador implementar límites de consumo de capacidad y recuento de objetos globales y específicos de la clase de almacenamiento por espacio de nombres. Además, usar un "límite de rango" garantiza que las solicitudes de PVC estén dentro de un valor mínimo y máximo antes de que la solicitud se reenvíe al provisionador.

Estos valores se definen por espacio de nombres, lo que significa que cada espacio de nombres debe tener valores definidos que se ajusten a sus necesidades de recursos. Mira aquí para obtener información sobre "cómo aprovechar las cuotas".

Configuración de almacenamiento

Cada plataforma de almacenamiento en el portafolio de NetApp tiene capacidades únicas que benefician a las aplicaciones, estén o no en contenedores.

Descripción general de la plataforma

Trident es compatible con ONTAP y Element. No existe una plataforma que se adapte mejor a todas las aplicaciones y escenarios que otra, sin embargo, al elegir una plataforma, se deben tener en cuenta las necesidades de la aplicación y del equipo que administra el dispositivo.

Deberías seguir las prácticas recomendadas básicas para el sistema operativo host con el protocolo que estás usando. Opcionalmente, podrías considerar incorporar las prácticas recomendadas de la aplicación, cuando estén disponibles, junto con la configuración de backend, storage class y PVC para optimizar el almacenamiento para aplicaciones específicas.

Mejores prácticas de ONTAP y Cloud Volumes ONTAP

Aprende las mejores prácticas para configurar ONTAP y Cloud Volumes ONTAP para Trident.

Las siguientes recomendaciones son pautas para configurar ONTAP para cargas de trabajo en contenedores que consumen volúmenes aprovisionados dinámicamente por Trident. Cada una debe considerarse y evaluarse para determinar su idoneidad en tu entorno.

Usa SVM dedicados a Trident

Las máquinas virtuales de almacenamiento (SVM) proporcionan aislamiento y separación administrativa entre los inquilinos en un sistema ONTAP. Dedicar una SVM a las aplicaciones permite la delegación de privilegios y permite aplicar las mejores prácticas para limitar el consumo de recursos.

Hay varias opciones disponibles para la gestión del SVM:

- Proporciona la interfaz de administración del clúster en la configuración del backend, junto con las credenciales adecuadas, y especifica el nombre de SVM.
- Crea una interfaz de administración dedicada para la SVM usando ONTAP System Manager o la CLI.
- Comparte el rol de administración con una interfaz de datos NFS.

En cada caso, la interfaz debe estar en DNS y el nombre DNS debe usarse al configurar Trident. Esto ayuda a facilitar algunos escenarios de recuperación ante desastres, por ejemplo, SVM-DR sin usar la retención de identidad de red.

No hay preferencia entre tener un LIF de administración dedicado o compartido para la SVM, pero debes asegurarte de que tus políticas de seguridad de red se alineen con el enfoque que elijas. De todas formas, el LIF de administración debe ser accesible mediante DNS para facilitar la máxima flexibilidad si "SVM-DR" se usa junto con Trident.

Limitar el recuento máximo de volúmenes

Los sistemas de almacenamiento ONTAP tienen un recuento máximo de volúmenes, que varía según la versión del software y la plataforma de hardware. Consulta "NetApp Hardware Universe" para tu plataforma específica y versión de ONTAP para determinar los límites exactos. Cuando se agota el recuento de volúmenes, las operaciones de aprovisionamiento fallan no solo para Trident, sino para todas las solicitudes de almacenamiento.

Los controladores de Trident `ontap-nas` y `ontap-san` aprovisionan un FlexVolume por cada volumen persistente (PV) de Kubernetes que se crea. El controlador `ontap-nas-economy` crea aproximadamente un FlexVolume por cada 200 PV (configurable entre 50 y 300). El controlador `ontap-san-economy` crea aproximadamente un FlexVolume por cada 100 PV (configurable entre 50 y 200). Para evitar que Trident consuma todos los volúmenes disponibles en el sistema de almacenamiento, deberías establecer un límite en la SVM. Puedes hacer esto desde la línea de comandos:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

El valor de `max-volumes` varía según varios criterios específicos de tu entorno:

- La cantidad de volúmenes existentes en el clúster de ONTAP
- La cantidad de volúmenes que esperas aprovisionar fuera de Trident para otras aplicaciones
- La cantidad de volúmenes persistentes que se espera que consuman las aplicaciones de Kubernetes

El `max-volumes` valor es la cantidad total de volúmenes aprovisionados en todos los nodos del clúster de ONTAP, y no en un nodo individual de ONTAP. Como resultado, podrías encontrarte con situaciones en las

que un nodo del clúster de ONTAP tenga muchos más o menos volúmenes aprovisionados con Trident que otro nodo.

Por ejemplo, un clúster ONTAP de dos nodos tiene la capacidad de alojar un máximo de 2000 FlexVol volúmenes. Tener el recuento máximo de volúmenes establecido en 1250 parece muy razonable. Sin embargo, si solo "agregados" de un nodo se asignan a la SVM, o los agregados asignados de un nodo no pueden aprovisionarse (por ejemplo, debido a la capacidad), entonces el otro nodo se convierte en el destino de todos los volúmenes aprovisionados por Trident. Esto significa que el límite de volúmenes podría alcanzarse para ese nodo antes de que se alcance el valor de `max-volumes`, lo que impacta tanto a Trident como a otras operaciones de volúmenes que usan ese nodo. **Puedes evitar esta situación asegurándote de que los agregados de cada nodo del clúster se asignen a la SVM que usa Trident en cantidades iguales.**

Clonar un volumen

NetApp Trident admite la clonación de volúmenes cuando se usan los `ontap-nas`, `ontap-san` y `solidfire-san` controladores de almacenamiento. Cuando se usan los `ontap-nas-flexgroup` o `ontap-nas-economy` controladores, la clonación no está soportada. Crear un volumen nuevo a partir de un volumen existente hará que se cree una nueva instantánea.

> Evita clonar un PVC que esté asociado con un StorageClass diferente. Realiza operaciones de clonación dentro del mismo StorageClass para garantizar la compatibilidad y evitar comportamientos inesperados.

Limita el tamaño máximo de los volúmenes creados por Trident

Para configurar el tamaño máximo para los volúmenes que puede crear Trident, usa el `limitVolumeSize` parámetro en tu `backend.json` definición.

Además de controlar el tamaño del volumen en la matriz de almacenamiento, también deberías aprovechar las capacidades de Kubernetes.

Limita el tamaño máximo de los FlexVols creados por Trident

Para configurar el tamaño máximo para los FlexVols usados como pools para los controladores ontap-san-economy y ontap-nas-economy, usa el parámetro `limitVolumePoolSize` en tu definición `backend.json`.

Configura Trident para usar CHAP bidireccional

Puedes especificar los nombres de usuario y las contraseñas del iniciador y del destino CHAP en la definición del backend y hacer que Trident habilite CHAP en la SVM. Usando el `useCHAP` parámetro en la configuración del backend, Trident autentica las conexiones iSCSI para backends ONTAP con CHAP.

Crear y usar una política de QoS de SVM

El uso de una política de QoS de ONTAP aplicada a la SVM limita la cantidad de IOPS consumibles por los volúmenes aprovisionados por Trident. Esto ayuda a "prevenir a un acosador"evitar que un contenedor fuera de control afecte las cargas de trabajo fuera de la SVM de Trident.

Puedes crear una política de QoS para la SVM en pocos pasos. Consulta la documentación de tu versión de ONTAP para la información más precisa. El siguiente ejemplo crea una política de QoS que limita el total de IOPS disponibles para la SVM a 5000.

```
# 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>
```

Además, si tu versión de ONTAP lo admite, puedes considerar usar un mínimo de QoS para garantizar una cantidad de ancho de banda para las cargas de trabajo en contenedores. La QoS adaptativa no es compatible con una política a nivel de SVM.

La cantidad de IOPS dedicadas a las cargas de trabajo en contenedores depende de muchos aspectos. Entre otras cosas, se incluyen:

- Otras cargas de trabajo que utilizan la matriz de almacenamiento. Si existen otras cargas de trabajo, no relacionadas con la implementación de Kubernetes, que utilizan los recursos de almacenamiento, se debe tener cuidado para asegurarse de que esas cargas de trabajo no se vean afectadas accidentalmente.
- Cargas de trabajo esperadas ejecutándose en contenedores. Si cargas de trabajo con altos requisitos de IOPS se ejecutan en contenedores, una política de QoS baja resulta en una mala experiencia.

Es importante recordar que una política de QoS asignada a nivel de SVM implica que todos los volúmenes aprovisionados en la SVM compartan el mismo grupo de IOPS. Si una o un pequeño número de aplicaciones contenedorizadas tienen un alto requerimiento de IOPS, podría convertirse en un obstáculo para las demás cargas de trabajo contenedorizadas. Si este es el caso, podrías considerar usar automatización externa para asignar políticas de QoS por volumen.

Deberías asignar el grupo de políticas QoS al SVM **solo** si tu versión de ONTAP es anterior a 9.8.

Crea grupos de políticas de QoS para Trident

La calidad de servicio (QoS) garantiza que el rendimiento de las cargas de trabajo críticas no se degrade por cargas de trabajo competidoras. Los grupos de políticas de QoS de ONTAP ofrecen opciones de QoS para volúmenes y permiten a los usuarios definir el límite de rendimiento para una o más cargas de trabajo. Para más información sobre QoS, consulta "Garantizar el rendimiento con QoS". Puedes especificar grupos de políticas de QoS en el backend o en un pool de almacenamiento, y se aplican a cada volumen creado en ese pool o backend.

ONTAP cuenta con dos tipos de grupos de políticas de QoS: tradicionales y adaptativos. Los grupos de políticas tradicionales proporcionan un máximo fijo (o mínimo, en versiones posteriores) de rendimiento en IOPS. La QoS adaptativa ajusta automáticamente el rendimiento al tamaño de la carga de trabajo, manteniendo la proporción de IOPS a TB o GB a medida que cambia el tamaño de la carga de trabajo. Esto proporciona una ventaja significativa cuando gestionas cientos o miles de cargas de trabajo en una implementación grande.

Ten en cuenta lo siguiente al crear grupos de políticas de QoS:

- Debes configurar la `qosPolicy`clave en el bloque `defaults` de la configuración del backend. Consulta el siguiente ejemplo de configuración del backend:

```
---
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
```

- Deberías aplicar los grupos de políticas por volumen, así cada volumen obtiene todo el rendimiento especificado por el grupo de políticas. No se admiten los grupos de políticas compartidos.

Para obtener más información sobre los grupos de políticas de QoS, consulta "Referencia de comandos de ONTAP".

Limita el acceso a los recursos de almacenamiento solo a los miembros del clúster de Kubernetes

Limitar el acceso a los volúmenes NFS, LUN iSCSI y LUN FC creados por Trident es un componente fundamental de la seguridad para tu implementación de Kubernetes. Hacer esto evita que los hosts que no forman parte del clúster de Kubernetes accedan a los volúmenes y puedan modificar los datos de forma inesperada.

Es importante comprender que los espacios de nombres son el límite lógico para los recursos en Kubernetes. Se asume que los recursos en el mismo espacio de nombres se pueden compartir, pero, lo importante es que no existe capacidad entre espacios de nombres. Esto significa que, aunque los PV son objetos globales, cuando se vinculan a un PVC solo son accesibles por pods que están en el mismo espacio de nombres. **Es fundamental asegurarse de que los espacios de nombres se usen para proporcionar separación cuando corresponde.**

La principal preocupación de la mayoría de las organizaciones con respecto a la seguridad de los datos en un contexto de Kubernetes es que un proceso en un contenedor pueda acceder al almacenamiento montado en el host, pero que no está destinado al contenedor. "Espacios de nombres" están diseñados para evitar este tipo de vulnerabilidad. Sin embargo, hay una excepción: los contenedores privilegiados.

Un contenedor privilegiado es aquel que se ejecuta con muchos más permisos a nivel de host de lo normal. Estos no se deniegan por defecto, así que asegúrate de deshabilitar la capacidad usando "políticas de seguridad de pod".

Para volúmenes a los que se desea acceder tanto desde Kubernetes como desde hosts externos, el

almacenamiento debe gestionarse de forma tradicional, con el PV introducido por el administrador y no gestionado por Trident. Esto asegura que el volumen de almacenamiento se destruya solo cuando tanto Kubernetes como los hosts externos se hayan desconectado y ya no estén usando el volumen. Además, se puede aplicar una política de exportación personalizada, que permite el acceso desde los nodos del clúster de Kubernetes y los servidores de destino fuera del clúster de Kubernetes.

Para implementaciones con nodos de infraestructura dedicados (por ejemplo, OpenShift) u otros nodos que no pueden programar aplicaciones de usuario, se deben usar políticas de exportación independientes para limitar aún más el acceso a los recursos de almacenamiento. Esto incluye crear una política de exportación para los servicios que se implementan en esos nodos de infraestructura (por ejemplo, los servicios de Métricas y Registro de OpenShift) y para las aplicaciones estándar que se implementan en nodos que no son de infraestructura.

Usa una política de exportación dedicada

Debes asegurarte de que exista una política de exportación para cada backend que solo permita el acceso a los nodos presentes en el clúster de Kubernetes. Trident puede crear y administrar políticas de exportación automáticamente. Así, Trident limita el acceso a los volúmenes que aprovisiona a los nodos del clúster de Kubernetes y simplifica la adición y eliminación de nodos.

Como alternativa, también puedes crear una política de exportación manualmente y llenarla con una o más reglas de exportación que procesen cada solicitud de acceso de nodo:

- Usa el `vserver export-policy create` comando CLI de ONTAP para crear la política de exportación.
- Agrega reglas a la política de exportación usando el comando CLI de `vserver export-policy rule create` ONTAP.

Al ejecutar estos comandos puedes restringir qué nodos de Kubernetes tienen acceso a los datos.

Deshabilita `showmount` para la aplicación SVM

La `showmount`función permite que un cliente de NFS consulte la SVM para obtener una lista de las exportaciones de NFS disponibles. Un pod implementado en el clúster de Kubernetes puede ejecutar el `showmount -e`comando contra la SVM y recibir una lista de los montajes disponibles, incluidos aquellos a los que no tiene acceso. Aunque esto por sí solo no es una vulnerabilidad de seguridad, sí proporciona información innecesaria que podría ayudar a un usuario no autorizado a conectarse a una exportación de NFS.

Debes deshabilitar `showmount` usando el comando de la línea de comandos de ONTAP a nivel SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire mejores prácticas

Conoce las mejores prácticas para configurar el almacenamiento SolidFire para Trident.

Crear cuenta de SolidFire

Cada cuenta de SolidFire representa a un propietario de volumen único y recibe su propio conjunto de credenciales de Challenge-Handshake Authentication Protocol (CHAP). Puedes acceder a los volúmenes asignados a una cuenta usando el nombre de la cuenta y las credenciales CHAP correspondientes o a través

de un grupo de acceso a volúmenes. Una cuenta puede tener hasta dos mil volúmenes asignados, pero un volumen solo puede pertenecer a una cuenta.

Crear una política de QoS

Usa las políticas de calidad de servicio (QoS) de SolidFire si quieres crear y guardar una configuración de calidad de servicio estandarizada que puedas aplicar a muchos volúmenes.

Puedes configurar los parámetros de QoS por volumen. El rendimiento de cada volumen se puede garantizar configurando tres parámetros configurables que definen la QoS: Min IOPS, Max IOPS y Burst IOPS.

Aquí tienes los posibles valores mínimos, máximos y de ráfaga de IOPS para el tamaño de bloque de 4Kb.

Parámetro IOPS	Definición	Valor mínimo	Valor predeterminado	Valor máximo (4Kb)
IOPS mínimas	El nivel de rendimiento garantizado para un volumen.	50	50	15000
IOPS máximo	El rendimiento no superará este límite.	50	15000	200.000
Burst IOPS	IOPS máximos permitidos en un escenario de ráfaga corta.	50	15000	200.000

> Aunque los IOPS máximos y los IOPS en ráfaga se pueden configurar hasta 200,000, el rendimiento máximo real de un volumen está limitado por el uso del clúster y el rendimiento por nodo.

El tamaño de bloque y el ancho de banda influyen directamente en el número de IOPS. A medida que aumenta el tamaño de bloque, el sistema aumenta el ancho de banda hasta el nivel necesario para procesar los tamaños de bloque más grandes. A medida que aumenta el ancho de banda, disminuye el número de IOPS que el sistema puede alcanzar. Consulta "Calidad de servicio de SolidFire" para más información sobre QoS y rendimiento.

Autenticación de SolidFire

Element admite dos métodos de autenticación: CHAP y Grupos de Acceso a Volumen (VAG). CHAP utiliza el protocolo CHAP para autenticar el host en el backend. Los Grupos de Acceso a Volumen controlan el acceso a los volúmenes que aprovisiona. NetApp recomienda usar CHAP para la autenticación, ya que es más sencillo y no tiene límites de escalabilidad.

> Trident con el aprovisionador CSI mejorado admite el uso de autenticación CHAP. Los VAG solo deben usarse en el modo de funcionamiento tradicional sin CSI.

La autenticación CHAP (verificación de que el iniciador es el usuario previsto del volumen) solo se admite con control de acceso basado en cuentas. Si usas CHAP para la autenticación, hay dos opciones disponibles: CHAP unidireccional y CHAP bidireccional. El CHAP unidireccional autentica el acceso al volumen usando el

nombre de la cuenta SolidFire y el secreto del iniciador. La opción CHAP bidireccional ofrece la forma más segura de autenticar el volumen porque el volumen autentica el host mediante el nombre de la cuenta y el secreto del iniciador, y luego el host autentica el volumen mediante el nombre de la cuenta y el secreto del destino.

Sin embargo, si no se puede habilitar CHAP y se requieren VAG, crea el grupo de acceso y añade los iniciadores de host y los volúmenes al grupo de acceso. Cada IQN que añades a un grupo de acceso puede acceder a cada volumen del grupo con o sin autenticación CHAP. Si el iniciador iSCSI está configurado para usar autenticación CHAP, se utiliza el control de acceso basado en cuentas. Si el iniciador iSCSI no está configurado para usar autenticación CHAP, entonces se utiliza el control de acceso del grupo de acceso a volúmenes.

¿Dónde encontrar más información?

A continuación, se incluye documentación sobre las mejores prácticas. Busca en "Biblioteca de NetApp" las versiones más recientes.

ONTAP

- "Guía de mejores prácticas e implementación de NFS"
- "Administración de SAN" (para iSCSI)
- "Configuración Express de iSCSI para RHEL"

Software Element

- "Configurar SolidFire para Linux"

NetApp HCI

- "Requisitos previos para la implementación de NetApp HCI"
- "Accede al NetApp Deployment Engine"

Información sobre las mejores prácticas de aplicación

- "Mejores prácticas para MySQL en ONTAP"
- "Prácticas recomendadas para MySQL en SolidFire"
- "NetApp SolidFire y Cassandra"
- "Prácticas recomendadas de Oracle en SolidFire"
- "Mejores prácticas de PostgreSQL en SolidFire"

No todas las aplicaciones tienen directrices específicas, es importante trabajar con tu equipo de NetApp y usar el "Biblioteca de NetApp" para encontrar la documentación más actualizada.

Integrar Trident

Para integrar Trident, los siguientes elementos de diseño y arquitectura requieren integración: selección e implementación de controladores, diseño de clases de almacenamiento, diseño de grupos virtuales, impactos de Persistent Volume Claim (PVC) en el aprovisionamiento de almacenamiento, operaciones de volumen y despliegue de servicios de OpenShift usando Trident.

Selección y despliegue de controladores

Selecciona e implementa un controlador de backend para tu sistema de almacenamiento.

Controladores de backend de ONTAP

Los controladores backend de ONTAP se diferencian por el protocolo utilizado y cómo se aprovisionan los volúmenes en el sistema de almacenamiento. Por lo tanto, piensa bien cuál controlador vas a implementar.

A un nivel superior, si tu aplicación tiene componentes que necesitan almacenamiento compartido (varios pods accediendo al mismo PVC), los controladores basados en NAS serían la opción predeterminada, mientras que los controladores iSCSI basados en bloques cubren las necesidades de almacenamiento no compartido. Elige el protocolo según los requisitos de la aplicación y el nivel de comodidad de los equipos de almacenamiento e infraestructura. En general, hay poca diferencia entre ellos para la mayoría de las aplicaciones, así que muchas veces la decisión se basa en si se necesita o no almacenamiento compartido (donde más de un pod necesitará acceso simultáneo).

Los controladores de backend ONTAP disponibles son:

- `ontap-nas`: Cada PV aprovisionado es un ONTAP completo FlexVolume.
- `ontap-nas-economy`: Cada PV aprovisionado es un qtree, con una cantidad configurable de qtrees por FlexVolume (el valor predeterminado es 200).
- `ontap-nas-flexgroup`: Cada PV se aprovisiona como un ONTAP FlexGroup completo, y se utilizan todos los agregados asignados a un SVM.
- `ontap-san`: cada PV aprovisionado es un LUN dentro de su propio FlexVolume.
- `ontap-san-economy`: cada PV aprovisionado es un LUN, con una cantidad configurable de LUN por FlexVolume (el valor predeterminado es 100).

Elegir entre los tres controladores NAS tiene algunas ramificaciones en las funciones que se ponen a disposición de la aplicación.

Ten en cuenta que, en las tablas a continuación, no todas las capacidades se exponen a través de Trident. Algunas deben ser aplicadas por el administrador de almacenamiento después del aprovisionamiento si se desea esa funcionalidad. Las notas al pie en superíndice distinguen la funcionalidad por característica y controlador.

Controladores ONTAP NAS	Instantáneas	Clones	Políticas dinámicas de exportación	Conexión múltiple	QoS	Cambiar tamaño	Replicación
`ontap-nas`	Sí	Sí	Sí [5]	Sí	Sí [1]	Sí	Sí [1]
`ontap-nas-economy`	NO [3]	NO [3]	Sí [5]	Sí	NO [3]	Sí	NO [3]
`ontap-nas-flexgroup`	Sí [1]	NO	Sí [5]	Sí	Sí [1]	Sí	Sí [1]

Trident ofrece 2 controladores SAN para ONTAP, cuyas capacidades se muestran a continuación.

Controladores SAN de ONTAP	Instantáneas	Clones	Conexión múltiple	CHAP bidireccional	QoS	Cambiar tamaño	Replicación
`ontap-san`	Sí	Sí	Sí nota al pie: 4[]	Sí	Sí [1]	Sí	Sí [1]
`ontap-san-economy`	Sí	Sí	Sí nota al pie: 4[]	Sí	NO [3]	Sí	NO [3]

> Nota al pie para las tablas anteriores: Sí [1]: No administrado por Trident Sí [2]: Administrado por Trident, pero no granular de PV NO [3]: No administrado por Trident y no granular de PV Sí [4]: Compatible con volúmenes raw-block Sí [5]: Compatible con Trident

Las características que no son granulares de PV se aplican a todo el FlexVolume y todos los PV (es decir, qtrees o LUNs en FlexVols compartidos) compartirán una programación común.

Como podemos ver en las tablas anteriores, gran parte de la funcionalidad entre el `ontap-nas` y el `ontap-nas-economy` es la misma. Sin embargo, porque el `ontap-nas-economy` driver limita la capacidad de controlar la programación con granularidad por PV, esto puede afectar tu planificación de recuperación ante desastres y copias de seguridad en particular. Para los equipos de desarrollo que quieren aprovechar la funcionalidad de clonado de PVC en almacenamiento ONTAP, esto solo es posible cuando usas los `ontap-nas`, `ontap-san` o `ontap-san-economy` drivers.

> El `solidfire-san`controlador también es capaz de clonar PVCs.

Controladores backend de Cloud Volumes ONTAP

Cloud Volumes ONTAP proporciona control de datos junto con funciones de almacenamiento para la gran empresa para varios casos de uso, incluyendo recursos compartidos de archivos y almacenamiento a nivel de bloque que sirven protocolos NAS y SAN (NFS, SMB / CIFS e iSCSI). Los controladores compatibles para Cloud Volume ONTAP son `ontap-nas`, `ontap-nas-economy`, `ontap-san` y `ontap-san-economy`. Estos son aplicables para Cloud Volume ONTAP para Azure, Cloud Volume ONTAP para GCP.

Controladores de backend de Amazon FSx for ONTAP

Amazon FSx for NetApp ONTAP te permite aprovechar las características, el rendimiento y las capacidades administrativas de NetApp que ya conoces, mientras aprovechas la simplicidad, agilidad, seguridad y escalabilidad de almacenar datos en AWS. FSx for ONTAP es compatible con muchas características del sistema de archivos ONTAP y APIs de administración. Los controladores compatibles para Cloud Volume ONTAP son `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` y `ontap-san-economy`.

NetApp HCI/SolidFire controladores backend

El `solidfire-san` controlador utilizado con las plataformas NetApp HCI/SolidFire ayuda al admin a configurar un backend de Element para Trident según los límites de QoS. Si quieres diseñar tu backend para establecer límites de QoS específicos en los volúmenes aprovisionados por Trident, usa el `type` parámetro en el archivo del backend. El admin también puede restringir el tamaño del volumen que se puede crear en el almacenamiento usando el `limitVolumeSize` parámetro. Actualmente, las funciones de almacenamiento de Element como el redimensionamiento y la replicación de volúmenes no están soportadas a través del

`solidfire-san` controlador. Estas operaciones deben hacerse manualmente desde la interfaz web de Element Software.

Controlador de SolidFire	Instantáneas	Clones	Conexión múltiple	CHAP	QoS	Cambiar tamaño	Replicación
`solidfire-san`	Sí	Sí	Sí	Sí	Sí	Sí	Sí [1]

> Nota al pie: Sínota al pie:1[]: No administrado por Trident SÍnota al pie:2[]: Compatible con volúmenes de bloques sin procesar

Controladores de backend de Azure NetApp Files

Trident utiliza el `azure-netapp-files` driver para gestionar el "Azure NetApp Files" service.

Más información sobre este controlador y cómo configurarlo la puedes encontrar en "Configuración del backend de Trident para Azure NetApp Files".

Controlador de archivos de Azure NetApp Files	Instantáneas	Clones	Conexión múltiple	QoS	Expandir	Replicación
`azure-netapp-files`	Sí	Sí	Sí	Sí	Sí	Sí [1]

> Nota al pie: Sínota al pie:1[]: No gestionado por Trident

Diseño de storage class

Es necesario configurar y aplicar clases de almacenamiento individuales para crear un objeto de clase de almacenamiento de Kubernetes. Esta sección explica cómo diseñar una clase de almacenamiento para tu aplicación.

Utilización específica del backend

El filtrado se puede usar dentro de un objeto de clase de almacenamiento específico para determinar qué grupo o conjunto de grupos de almacenamiento se usará con esa clase de almacenamiento específica. Se pueden configurar tres conjuntos de filtros en la clase de almacenamiento: `storagePools`, `additionalStoragePools`, y/o `excludeStoragePools`.

El `storagePools` parámetro ayuda a restringir el almacenamiento al conjunto de pools que coinciden con cualquier atributo especificado. El `additionalStoragePools` parámetro se usa para ampliar el conjunto de pools que Trident utiliza para el aprovisionamiento junto con el conjunto de pools seleccionados por los atributos y los parámetros `storagePools`. Puedes usar cualquiera de los parámetros por separado o ambos juntos para asegurarte de que se seleccione el conjunto adecuado de pools de almacenamiento.

El `excludeStoragePools`parámetro se usa para excluir específicamente el conjunto de pools que coinciden con los atributos.

Emular políticas de QoS

Si quieres diseñar Storage Classes para emular políticas de Quality of Service, crea una Storage Class con el `media` atributo como `hdd` o `ssd`. Según el `media` atributo mencionado en la storage class, Trident seleccionará el backend adecuado que sirve `hdd` o `ssd` aggregates para que coincida con el atributo media y luego dirigirá el aprovisionamiento de los volúmenes al agregado específico. Por eso, podemos crear una

storage class PREMIUM que tenga el `media` atributo establecido como `ssd`, lo que podría clasificarse como la política de QoS PREMIUM. Podemos crear otra storage class STANDARD que tenga el atributo media establecido como `hdd`, lo que podría clasificarse como la política de QoS STANDARD. También podemos usar el atributo ``IOPS'' en la storage class para redirigir el aprovisionamiento a un Element appliance, que puede definirse como una política de QoS.

Utiliza el backend en función de características específicas

Las clases de almacenamiento se pueden diseñar para dirigir el aprovisionamiento de volúmenes en un backend específico donde se habilitan funciones como aprovisionamiento fino y grueso, instantáneas, clones y cifrado. Para especificar qué almacenamiento usar, crea clases de almacenamiento que especifiquen el backend adecuado con la función requerida habilitada.

Pools virtuales

Los grupos virtuales están disponibles para todos los backends de Trident. Puedes definir grupos virtuales para cualquier backend usando cualquier controlador que Trident proporciona.

Los pools virtuales permiten a un administrador crear un nivel de abstracción sobre los backends que se puede referenciar mediante Storage Classes, para mayor flexibilidad y una colocación eficiente de volúmenes en los backends. Se pueden definir diferentes backends con la misma clase de servicio. Además, se pueden crear múltiples storage pools en el mismo backend pero con diferentes características. Cuando una Storage Class se configura con un selector con etiquetas específicas, Trident elige un backend que coincida con todas las etiquetas del selector para colocar el volumen. Si las etiquetas del selector de la Storage Class coinciden con varios storage pools, Trident elegirá uno de ellos para aprovisionar el volumen.

Diseño de pool virtual

Al crear un backend, generalmente puedes especificar un conjunto de parámetros. Era imposible para el administrador crear otro backend con las mismas credenciales de almacenamiento y con un conjunto diferente de parámetros. Con la introducción de los pools virtuales, este problema se ha aliviado. Un pool virtual es una abstracción de nivel introducida entre el backend y la Kubernetes Storage Class para que el administrador pueda definir parámetros junto con etiquetas que pueden ser referenciadas a través de las Kubernetes Storage Classes como un selector, de una manera independiente del backend. Se pueden definir pools virtuales para todos los backends compatibles de NetApp con Trident. Esa lista incluye SolidFire/NetApp HCI, ONTAP, así como Azure NetApp Files.

> Al definir grupos virtuales, se recomienda no reorganizar el orden de los grupos virtuales existentes en una definición de backend. También es recomendable no editar ni modificar los atributos de un grupo virtual existente y definir un nuevo grupo virtual en su lugar.

Emulando diferentes niveles de servicio/QoS

Es posible diseñar grupos virtuales para emular clases de servicio. Usando la implementación de grupo virtual de Cloud Volume Service para Azure NetApp Files, veamos cómo podemos configurar diferentes clases de servicio. Configura el backend de Azure NetApp Files con múltiples etiquetas que representen diferentes niveles de rendimiento. Establece el aspecto `servicelevel` en el nivel de rendimiento adecuado y agrega otros aspectos necesarios bajo cada etiqueta. Ahora crea diferentes clases de almacenamiento de Kubernetes que se asignarán a diferentes grupos virtuales. Usando el campo `parameters.selector`, cada StorageClass indica qué grupos virtuales se pueden usar para alojar un volumen.

Asignar un conjunto específico de aspectos

Se pueden diseñar varios pools virtuales con un conjunto específico de aspectos desde un único backend de almacenamiento. Para hacerlo, configura el backend con varias etiquetas y establece los aspectos necesarios bajo cada etiqueta. Ahora crea diferentes clases de almacenamiento de Kubernetes usando el campo `parameters.selector` que se asignará a distintos pools virtuales. Los volúmenes que se aprovisionen en el backend tendrán los aspectos definidos en el pool virtual elegido.

Características del PVC que afectan el aprovisionamiento de almacenamiento

Algunos parámetros más allá de la clase de almacenamiento solicitada pueden afectar el proceso de decisión de aprovisionamiento de Trident al crear un PVC.

Modo de acceso

Cuando solicitas almacenamiento a través de un PVC, uno de los campos obligatorios es el modo de acceso. El modo que elijas puede afectar el backend seleccionado para alojar la solicitud de almacenamiento.

Trident intentará hacer coincidir el protocolo de almacenamiento utilizado con el método de acceso especificado según la siguiente matriz. Esto es independiente de la plataforma de almacenamiento subyacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	Sí	Sí	Sí (Raw block)
NFS	Sí	Sí	Sí

Una solicitud de un PVC de ReadWriteMany enviada a una implementación de Trident sin un backend NFS configurado dará como resultado que no se aprovisione ningún volumen. Por esta razón, el solicitante debe usar el modo de acceso que sea apropiado para su aplicación.

Operaciones de volumen

Modificar volúmenes persistentes

Los volúmenes persistentes son, con dos excepciones, objetos inmutables en Kubernetes. Una vez creados, la política de recuperación y el tamaño se pueden modificar. Sin embargo, esto no impide que algunos aspectos del volumen se modifiquen fuera de Kubernetes. Esto puede ser deseable para personalizar el volumen para aplicaciones específicas, para asegurarte de que la capacidad no se consuma accidentalmente o simplemente para mover el volumen a un controlador de almacenamiento diferente por cualquier motivo.

> Por el momento, los provisionadores en árbol de Kubernetes no admiten operaciones de redimensionamiento de volúmenes para NFS, iSCSI o FC PV. Trident admite la expansión de volúmenes NFS, iSCSI y FC.

Los detalles de conexión del PV no se pueden modificar después de la creación.

Crea instantáneas de volumen bajo demanda

Trident admite la creación de instantáneas de volumen bajo demanda y la creación de PVCs a partir de instantáneas mediante el framework CSI. Las instantáneas proporcionan un método práctico para mantener copias de un momento específico de los datos y tienen un ciclo de vida independiente del PV de origen en Kubernetes. Estas instantáneas se pueden usar para clonar PVCs.

Crear volúmenes a partir de instantáneas

Trident también admite la creación de PersistentVolumes a partir de instantáneas de volúmenes. Para lograr esto, solo tienes que crear un PersistentVolumeClaim y mencionar el `datasource` como la instantánea requerida desde la cual se debe crear el volumen. Trident gestionará este PVC creando un volumen con los datos presentes en la instantánea. Con esta función, puedes duplicar datos entre regiones, crear entornos de prueba, reemplazar un volumen de producción dañado o corrupto por completo, o recuperar archivos y directorios específicos y transferirlos a otro volumen adjunto.

Mover volúmenes en el clúster

Los administradores de almacenamiento tienen la capacidad de mover volúmenes entre agregados y controladores en el clúster ONTAP de forma no disruptiva para el consumidor de almacenamiento. Esta operación no afecta a Trident ni al clúster Kubernetes, siempre y cuando el agregado de destino sea uno al que la SVM que usa Trident tenga acceso. Lo importante es que, si el agregado se ha añadido recientemente a la SVM, será necesario actualizar el backend volviéndolo a agregar a Trident. Esto hará que Trident vuelva a inventariar la SVM para que se reconozca el nuevo agregado.

Sin embargo, mover volúmenes entre backends no es compatible automáticamente por Trident. Esto incluye entre SVMs en el mismo clúster, entre clústeres o en una plataforma de almacenamiento diferente (incluso si ese sistema de almacenamiento está conectado a Trident).

Si se copia un volumen a otra ubicación, se puede usar la función de importación de volúmenes para importar los volúmenes actuales en Trident.

Ampliar volúmenes

Trident admite el cambio de tamaño de NFS, iSCSI y FC PVs. Esto permite a los usuarios cambiar el tamaño de sus volúmenes directamente a través de la capa de Kubernetes. La expansión del volumen es posible para todas las principales plataformas de almacenamiento NetApp, incluyendo ONTAP y backends SolidFire/NetApp HCI. Para permitir una posible expansión después, establece `allowVolumeExpansion` en `true` en tu StorageClass asociado con el volumen. Cuando sea necesario cambiar el tamaño del volumen persistente, edita la anotación `spec.resources.requests.storage` en la Persistent Volume Claim al tamaño de volumen requerido. Trident se encargará automáticamente de cambiar el tamaño del volumen en el clúster de almacenamiento.

Importa un volumen existente en Kubernetes

La importación de volúmenes ofrece la posibilidad de importar un volumen de almacenamiento existente en un entorno Kubernetes. Esto es compatible actualmente con los controladores `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` y `azure-netapp-files`. Esta función es útil cuando portas una aplicación existente a Kubernetes o durante escenarios de recuperación ante desastres.

Cuando uses los controladores ONTAP y `solidfire-san` utiliza el comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` para importar un volumen existente en Kubernetes para que sea gestionado por Trident. El archivo YAML o JSON de PVC usado en el comando de importación de volumen apunta a una clase de almacenamiento que identifica a Trident como el provisioner. Cuando uses un backend NetApp HCI/SolidFire, asegúrate de que los nombres de los volúmenes sean únicos. Si los nombres de los volúmenes están duplicados, clona el volumen con un nombre único para que la función de importación de volumen pueda distinguir entre ellos.

Si se utiliza el controlador `azure-netapp-files`, usa el comando `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` para importar el volumen en Kubernetes y que sea gestionado por Trident. Esto garantiza una referencia de volumen única.

Cuando se ejecute el comando anterior, Trident encontrará el volumen en el backend y leerá su tamaño. Añadirá automáticamente (y sobrescribirá si es necesario) el tamaño de volumen del PVC configurado. Luego, Trident crea el nuevo PV y Kubernetes vincula el PVC al PV.

Si un contenedor fue desplegado de tal manera que requería el PVC importado específico, permanecería en estado pendiente hasta que el par PVC/PV se vincule mediante el proceso de importación de volumen. Después de que el par PVC/PV se vincule, el contenedor debería iniciarse, siempre que no haya otros problemas.

Servicio de registro

El despliegue y la gestión del almacenamiento para el registro se han documentado en "netapp.io" en el "blog".

Servicio de registro

Al igual que otros servicios de OpenShift, el servicio de registro se implementa usando Ansible con parámetros de configuración suministrados por el archivo de inventario, o sea, hosts, proporcionado al playbook. Hay dos métodos de instalación que se van a cubrir: implementar el registro durante la instalación inicial de OpenShift e implementar el registro después de que OpenShift ya esté instalado.

A partir de la versión 3.9 de Red Hat OpenShift, la documentación oficial desaconseja NFS para el servicio de registro debido a preocupaciones sobre la corrupción de datos. Esto se basa en pruebas de Red Hat con sus productos. El servidor NFS de ONTAP no tiene estos problemas y puede respaldar fácilmente un despliegue de registro. Al final, la elección del protocolo para el servicio de registro depende de ti, solo ten en cuenta que ambos funcionarán genial cuando uses plataformas NetApp y no hay razón para evitar NFS si esa es tu preferencia.

Si decides usar NFS con el servicio de registro, tendrás que establecer la variable de Ansible `openshift_enable_unsupported_configurations` en `true` para evitar que el instalador falle.

Empezar

El servicio de registro puede, opcionalmente, desplegarse tanto para aplicaciones como para las operaciones principales del clúster OpenShift en sí. Si decides desplegar el registro de operaciones, especificando la variable `openshift_logging_use_ops` como `true`, se crearán dos instancias del servicio. Las variables que controlan la instancia de registro para operaciones contienen "ops", mientras que la instancia para aplicaciones no.

Configurar las variables de Ansible según el método de despliegue es importante para garantizar que los servicios subyacentes utilicen el almacenamiento correcto. Veamos las opciones para cada uno de los métodos de despliegue.

Las tablas siguientes contienen únicamente las variables relevantes para la configuración del almacenamiento en lo que respecta al servicio de registro. Puedes encontrar otras opciones en "Documentación de registro de Red Hat OpenShift" que deberías revisar, configurar y usar según tu implementación.

Las variables de la tabla siguiente harán que el libro de jugadas de Ansible cree un PV y un PVC para el servicio de registro usando los detalles proporcionados. Este método es significativamente menos flexible que usar el libro de jugadas de instalación de componentes después de la instalación de OpenShift, pero si tienes volúmenes existentes disponibles, es una opción.

Variable	Detalles
`openshift_logging_storage_kind`	Establécelo en `nfs` para que el instalador cree un NFS PV para el servicio de registro.
`openshift_logging_storage_host`	El nombre de host o la dirección IP del host NFS. Esto debe configurarse en el dataLIF de tu máquina virtual.
`openshift_logging_storage_nfs_directory`	La ruta de montaje para la exportación NFS. Por ejemplo, si el volumen está unido como `/openshift_logging`, usarías esa ruta para esta variable.
`openshift_logging_storage_volume_name`	El nombre, por ejemplo `pv_ose_logs`, del PV a crear.
`openshift_logging_storage_volume_size`	El tamaño de la exportación NFS, por ejemplo `100Gi`.

Si tu clúster de OpenShift ya está funcionando y, por lo tanto, Trident ha sido desplegado y configurado, el instalador puede usar el aprovisionamiento dinámico para crear los volúmenes. Las siguientes variables deberán configurarse.

Variable	Detalles
`openshift_logging_es_pvc_dynamic`	Establécelo en true para usar volúmenes aprovisionados dinámicamente.
`openshift_logging_es_pvc_storage_class_name`	El nombre de la clase de almacenamiento que se utilizará en el PVC.
`openshift_logging_es_pvc_size`	El tamaño del volumen solicitado en el PVC.
`openshift_logging_es_pvc_prefix`	Un prefijo para los PVC usados por el servicio de registro.
`openshift_logging_es_ops_pvc_dynamic`	Establécelo en `true` para usar volúmenes aprovisionados dinámicamente para la instancia de registro de operaciones.
`openshift_logging_es_ops_pvc_storage_class_name`	El nombre de la clase de almacenamiento para la instancia de registro ops.
`openshift_logging_es_ops_pvc_size`	El tamaño de la solicitud de volumen para la instancia ops.
`openshift_logging_es_ops_pvc_prefix`	Un prefijo para los PVC de la instancia ops.

Despliega la pila de registro

Si estás desplegando el registro como parte del proceso de instalación inicial de OpenShift, solo necesitas seguir el proceso de despliegue estándar. Ansible configurará y desplegará los servicios necesarios y los objetos de OpenShift para que el servicio esté disponible en cuanto Ansible termine.

Sin embargo, si estás haciendo el despliegue después de la instalación inicial, Ansible deberá usar el playbook del componente. Este proceso puede cambiar ligeramente con diferentes versiones de OpenShift, así que asegúrate de leer y seguir "Red Hat OpenShift Container Platform 3.11 documentación" para tu versión.

Servicio de métricas

El servicio de métricas proporciona información valiosa al administrador sobre el estado, la utilización de recursos y la disponibilidad del clúster OpenShift. También es necesario para la funcionalidad de autoescalado de pods y muchas organizaciones usan datos del servicio de métricas para sus aplicaciones de charge back y/o show back.

Al igual que con el servicio de registro, y OpenShift en su conjunto, Ansible se usa para desplegar el servicio de métricas. También, como el servicio de registro, el servicio de métricas se puede desplegar durante una configuración inicial del clúster o después de que esté en funcionamiento usando el método de instalación de componentes. Las siguientes tablas contienen las variables que son importantes al configurar el almacenamiento persistente para el servicio de métricas.

Las tablas siguientes solo contienen las variables relevantes para la configuración del almacenamiento en relación con el servicio de métricas. Hay muchas otras opciones en la documentación que deberías revisar, configurar y usar según tu puesta en marcha.

Variable	Detalles
`openshift_metrics_storage_kind`	Establécelo en `nfs` para que el instalador cree un NFS PV para el servicio de registro.
`openshift_metrics_storage_host`	El nombre de host o la dirección IP del host NFS. Esto debe establecerse en el dataLIF de tu SVM.
`openshift_metrics_storage_nfs_directory`	La ruta de montaje para la exportación NFS. Por ejemplo, si el volumen está unido como `/openshift_metrics`, usarías esa ruta para esta variable.
`openshift_metrics_storage_volume_name`	El nombre, por ejemplo `pv_ose_metrics`, del PV a crear.
`openshift_metrics_storage_volume_size`	El tamaño de la exportación NFS, por ejemplo `100Gi`.

Si tu clúster de OpenShift ya está funcionando y, por lo tanto, Trident ha sido desplegado y configurado, el instalador puede usar el aprovisionamiento dinámico para crear los volúmenes. Las siguientes variables deberán configurarse.

Variable	Detalles
`openshift_metrics_cassandra_pvc_prefix`	Un prefijo que se usará para los PVC de métricas.
`openshift_metrics_cassandra_pvc_size`	El tamaño de los volúmenes a solicitar.
`openshift_metrics_cassandra_storage_type`	El tipo de almacenamiento que se debe usar para las métricas, esto debe estar configurado como dinámico para que Ansible cree PVC con la clase de almacenamiento adecuada.
`openshift_metrics_cassanda_pvc_storage_class_name`	El nombre de la clase de almacenamiento que se va a usar.

Implementa el servicio de métricas

Con las variables Ansible apropiadas definidas en tu archivo hosts/inventory, despliega el servicio usando

Ansible. Si estás desplegando en el momento de instalación de OpenShift, entonces el PV se creará y usará automáticamente. Si estás desplegando usando los playbooks de componentes, después de la instalación de OpenShift, entonces Ansible crea cualquier PVC que sea necesario y, después de que Trident haya aprovisionado almacenamiento para ellos, despliega el servicio.

Las variables anteriores y el proceso de despliegue pueden cambiar con cada versión de OpenShift. Asegúrate de revisar y seguir "Guía de puesta en marcha de OpenShift de Red Hat" para tu versión, así estará configurado para tu entorno.

Protección de datos y recuperación de desastres

Conoce las opciones de protección y recuperación para Trident y los volúmenes creados usando Trident. Deberías tener una estrategia de protección y recuperación de datos para cada aplicación con un requisito de persistencia.

Replicación y recuperación de Trident

Puedes crear una copia de seguridad para restaurar Trident en caso de desastre.

Replicación de Trident

Trident utiliza los CRD de Kubernetes para almacenar y gestionar su propio estado y el etcd del clúster de Kubernetes para almacenar sus metadatos.

Pasos

1. Haz una copia de seguridad del clúster Kubernetes etcd usando "Kubernetes: copia de seguridad de un clúster etcd".
2. Coloca los artefactos de copia de seguridad en un volumen FlexVol

> NetApp recomienda que protejas la SVM donde reside el FlexVol con una relación de SnapMirror hacia otra SVM.

Recuperación de Trident

Usando los CRD de Kubernetes y la instantánea etcd del clúster de Kubernetes, puedes recuperar Trident.

Pasos

1. Desde la SVM de destino, monta el volumen que contiene los archivos de datos etcd de Kubernetes y los certificados en el host que se configurará como nodo maestro.
2. Copia todos los certificados necesarios relacionados con el clúster de Kubernetes en `/etc/kubernetes/pki` y los archivos de miembros de etcd en `/var/lib/etcd`.
3. Restaura el clúster de Kubernetes desde la copia de seguridad de etcd usando "Kubernetes: restaurar un clúster etcd".
4. Ejecuta `kubectl get crd` para verificar que todos los recursos personalizados de Trident han aparecido y recupera los objetos Trident para comprobar que todos los datos están disponibles.

Replicación y recuperación de SVM

Trident no puede configurar las relaciones de replicación, sin embargo, el administrador de almacenamiento

puede usar "ONTAP SnapMirror" para replicar una SVM.

En caso de desastre, puedes activar la SVM de destino SnapMirror para empezar a servir datos. Puedes volver al primario cuando se restauren los sistemas.

Acerca de esta tarea

Ten en cuenta lo siguiente cuando uses la función SnapMirror SVM Replication:

- Deberías crear un backend distinto para cada SVM con SVM-DR activado.
- Configura las clases de almacenamiento para seleccionar los backends replicados solo cuando sea necesario, para evitar tener volúmenes que no necesitan replicación aprovisionados en los backends que soportan SVM-DR.
- Los administradores de aplicaciones deben comprender el coste y la complejidad adicionales asociados con la replicación y considerar detenidamente su plan de recuperación antes de comenzar este proceso.

Replicación SVM

Puedes usar "ONTAP: SnapMirror replicación SVM" para crear la relación de replicación SVM.

SnapMirror te permite establecer opciones para controlar qué se va a replicar. Vas a necesitar saber qué opciones seleccionaste al realizar Recuperación de SVM usando Trident.

- "-identity-preserve true" replica toda la configuración del SVM.
- "-discard-configs red" excluye las LIF y la configuración de red relacionada.
- "-identity-preserve false" replica solo los volúmenes y la configuración de seguridad.

Recuperación de SVM usando Trident

Trident no detecta automáticamente los fallos de la SVM. En caso de desastre, el administrador puede iniciar manualmente la conmutación por error de Trident a la nueva SVM.

Pasos

1. Cancela las transferencias programadas y en curso de SnapMirror, rompe la relación de replicación, detén la SVM de origen y luego activa la SVM de destino de SnapMirror.
2. Si especificaste `-identity-preserve false` o `-discard-config network` al configurar tu replicación de SVM, actualiza el `managementLIF` y el `dataLIF` en el archivo de definición del backend de Trident.
3. Confirma que `storagePrefix` está presente en el archivo de definición del backend de Trident. Este parámetro no se puede cambiar. Omitir `storagePrefix` hará que la actualización del backend falle.
4. Actualiza todos los backends necesarios para reflejar el nuevo nombre de SVM de destino usando:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Si especificaste `-identity-preserve false` o `discard-config network`, tienes que reiniciar todos los pods de la aplicación.

Si especificaste `-identity-preserve true`, todos los volúmenes aprovisionados por Trident comienzan a servir datos cuando se activa la SVM de destino.

Replicación y recuperación de volúmenes

Trident no puede configurar las relaciones de replicación de SnapMirror, sin embargo, el administrador de almacenamiento puede usar "Replicación y recuperación de ONTAP SnapMirror" para replicar volúmenes creados por Trident.

Luego, puedes importar los volúmenes recuperados en Trident usando "tridentctl volume import".

La importación no es compatible en `ontap-nas-economy`, `ontap-san-economy` o en `ontap-flexgroup-economy` drivers.

Protección de datos Snapshot

Puedes proteger y restaurar datos usando:

- Un controlador de instantáneas externo y CRDs para crear instantáneas de volúmenes de Kubernetes de Volúmenes Persistentes (PVs).

  "Instantáneas de volumen"

- ONTAP Snapshots para restaurar todo el contenido de un volumen o recuperar archivos individuales o LUNs.

  "Instantáneas de ONTAP"

Automatizando la conmutación por error de aplicaciones con estado con Trident

La función de separación forzada de Trident te permite separar automáticamente volúmenes de nodos insalubres en un clúster de Kubernetes, evitando la corrupción de datos y asegurando la disponibilidad de las aplicaciones. Esta función es especialmente útil en escenarios donde los nodos dejan de responder o se desconectan para mantenimiento.

Detalles sobre la desconexión forzada

Force detach está disponible para `ontap-san`, `ontap-san-economy`, `ontap-nas` y `ontap-nas-economy` solamente. Antes de activar force detach, debe estar habilitado el apagado no pacífico de nodos (NGNS) en el clúster de Kubernetes. NGNS está habilitado por defecto para Kubernetes 1.28 y versiones superiores. Para más información, consulta "Kubernetes: apagado no controlado del nodo".

Cuando usas el controlador `ontap-nas` o `ontap-nas-economy`, necesitas establecer el parámetro `autoExportPolicy` en la configuración del backend a `true` para que Trident pueda restringir el acceso desde el nodo de Kubernetes con el taint aplicado usando políticas de exportación gestionadas.

Debido a que Trident depende de Kubernetes NGNS, no elimines `out-of-service` taints de un nodo no saludable hasta que todas las cargas de trabajo no tolerables se hayan reprogramado. Aplicar o eliminar el taint de forma imprudente puede poner en riesgo la protección de datos del backend.

Cuando el administrador del clúster de Kubernetes ha aplicado la `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` mancha al nodo y `enableForceDetach` está configurado en `true`, Trident determinará el estado del nodo y:

1. Detén el acceso de E/S del backend para los volúmenes montados en ese nodo.
2. Marca el objeto nodo Trident como `dirty` (no seguro para nuevas publicaciones).

El controlador Trident rechazará nuevas solicitudes de publicación de volumen hasta que el nodo sea recalificado (después de haber sido marcado como `dirty`) por el pod de nodo Trident. Cualquier carga de trabajo programada con un PVC montado (incluso después de que el nodo del clúster esté en buen estado y listo) no se aceptará hasta que Trident pueda verificar el nodo `clean` (seguro para nuevas publicaciones).

Cuando se restablezca la salud del nodo y se elimine la taint, Trident hará lo siguiente:

1. Identifica y limpia las rutas publicadas obsoletas en el nodo.
2. Si el nodo está en un `cleanable` estado (se ha eliminado la mancha fuera de servicio y el nodo está en `Ready` estado) y todas las rutas publicadas obsoletas están limpias, Trident readmitirá el nodo como `clean` y permitirá nuevos volúmenes publicados en el nodo.

Detalles sobre la conmutación automática al respaldo

Puedes automatizar el proceso de desconexión forzada mediante la integración con "operador de verificación de estado del nodo (NHC)". Cuando ocurre un fallo en un nodo, NHC activa la remediación de nodos Trident (TNR) y la desconexión forzada automáticamente creando un CR TridentNodeRemediation en el espacio de nombres de Trident que define el nodo fallido. TNR se crea solo cuando falla un nodo y NHC la elimina una vez que el nodo vuelve a estar en línea o se elimina.

Proceso de eliminación de pod de nodo con fallo

La conmutación automática al respaldo selecciona las cargas de trabajo que se eliminarán del nodo fallido. Cuando se crea un TNR, el controlador del TNR marca el nodo como sucio, impide la publicación de nuevos volúmenes y comienza a eliminar los pods compatibles con la desconexión forzada y sus conexiones de volumen.

Todos los volúmenes/PVC compatibles con force-detach son compatibles con la conmutación automática al respaldo:

- Volúmenes NAS y NAS-economy que utilizan políticas de exportación automática (SMB aún no es compatible).
- Volúmenes SAN y SAN-economy.

Consulta Detalles sobre la desconexión forzada.

Comportamiento predeterminado:

- Los pods que utilizan volúmenes compatibles con la desconexión forzada se eliminan del nodo fallido. Kubernetes volverá a programarlos en un nodo en buen estado.
- Los pods que usan un volumen no compatible con la desconexión forzada, incluidos los volúmenes que no son Trident, no se eliminan del nodo fallido.
- Los pods sin estado (no PVC) no se eliminan del nodo fallido, a menos que la anotación del pod `trident.netapp.io/podRemediationPolicy: delete` esté configurada.

Anulación del comportamiento de eliminación de pod:

El comportamiento de eliminación de pods se puede personalizar usando una anotación de pod: `trident.netapp.io/podRemediationPolicy[retain, delete]`. Estas anotaciones se examinan y se usan cuando ocurre una conmutación al respaldo. Aplica anotaciones a la especificación del pod en el deployment o replicaset de Kubernetes para evitar que la anotación desaparezca después de una conmutación al respaldo:

- `retain` - El pod NO se eliminará del nodo fallido durante una conmutación automática al respaldo.
- `delete` - El pod se eliminará del nodo fallido durante una conmutación automática al respaldo.

Estas anotaciones se pueden aplicar a cualquier pod.

- Las operaciones de E/S solo se bloquearán en los nodos fallidos para los volúmenes que admiten force-detach.
- Para los volúmenes que no admiten la desconexión forzada, existe el riesgo de corrupción de datos y problemas de multi-attach.

CR de TridentNodeRemediation

El CR TridentNodeRemediation (TNR) define un nodo fallido. El nombre del TNR es el nombre del nodo fallido.

Ejemplo TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

Estados de TNR: usa los siguientes comandos para ver el estado de los TNR:
`kubectl get tnr <name> -n <trident-namespace>`

Los TNR pueden estar en uno de los siguientes estados:

- *Remediando*:
  - Detén el acceso de E/S de backend para los volúmenes compatibles con force-detach que están montados en ese nodo.
  - El objeto de nodo Trident está marcado como sucio (no es seguro para nuevas publicaciones).
  - Eliminar pods y adjuntos de volumen del nodo

- *Recuperación de nodo pendiente*:
  - El controlador está esperando que el nodo vuelva a estar en línea.
  - Una vez que el nodo esté en línea, publish-enforcement se asegurará de que el nodo esté limpio y listo para nuevas publicaciones de volúmenes.
- Si se elimina el nodo de K8s, el controlador TNR eliminará el TNR y dejará de conciliar.
- *Correcto*:
  - Todos los pasos de remediación y recuperación del nodo se completaron correctamente. El nodo está limpio y listo para nuevas publicaciones de volúmenes.
- *Fallido*:
  - Error irrecuperable. Las causas del error se especifican en el campo status.message del CR.

Habilitar la conmutación automática al respaldo

Prerrequisitos:

- Asegúrate de que la desconexión forzada esté habilitada antes de habilitar la conmutación automática al respaldo. Para más información, consulta Detalles sobre la desconexión forzada.
- Instala la comprobación del estado del nodo (NHC) en el clúster de Kubernetes.
  - "Instala operator-sdk".
  - Instala Operator Lifecycle Manager (OLM) en el clúster si aún no está instalado: `operator-sdk olm install`.
  - Instala Node Health check Operator: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`.

> También puedes usar formas alternativas para detectar fallas de nodos como se especifica en la sección de [Integrating Custom Node Health Check Solutions] abajo.

Consulta "Operador de comprobación de estado del nodo" para más información.

Pasos

1. Crea un NodeHealthCheck (NHC) en el espacio de nombres Trident para monitorear los nodos de trabajo en el clúster. Ejemplo:

```
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. Aplica la verificación de estado del nodo CR en el `trident` namespace.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

El CR anterior está configurado para supervisar los nodos de trabajo de K8s en busca de las condiciones Ready: false y Unknown. La conmutación automática al respaldo se activará cuando un nodo entre en estado Ready: false o Ready: Unknown.

El `unhealthyConditions` en el CR usa un periodo de gracia de 0 segundos. Esto hace que la conmutación automática al respaldo se active de inmediato cuando K8s establece la condición del nodo Ready: false, que se configura después de que K8s pierde el latido de un nodo. K8s tiene un valor predeterminado de 40 segundos de espera después del último latido antes de establecer Ready: false. Este periodo de gracia se puede personalizar en las opciones de despliegue de K8s.

Para obtener opciones de configuración adicionales, consulta "Documentación de Node-Healthcheck-Operator".

Información adicional de configuración

Cuando se instala Trident con force-detach habilitado, se crean automáticamente dos recursos adicionales en el espacio de nombres de Trident para facilitar la integración con NHC: TridentNodeRemediationTemplate (TNRT) y ClusterRole.

TridentNodeRemediationTemplate (TNRT):

El TNRT sirve como plantilla para el controlador del NHC, que usa TNRT para generar recursos TNR según sea necesario.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole:

También se agrega un rol de clúster durante la instalación cuando se habilita la desconexión forzada. Esto le da a NHC permisos para los TNR en el espacio de nombres Trident.

```
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
```

Actualizaciones y mantenimiento del clúster K8s

Para evitar cualquier conmutación automática al respaldo, pausa la conmutación automática al respaldo durante el mantenimiento o las actualizaciones de K8s, cuando se espera que los nodos se apaguen o se reinicien. Puedes pausar el CR de NHC (descrito arriba) parchando su CR:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Esto pausa la conmutación automática al respaldo. Para volver a habilitar la conmutación automática al respaldo, elimina pauseRequests de la especificación después de que termine el mantenimiento.

Limitaciones

- Las operaciones de E/S solo se impiden en los nodos fallidos para los volúmenes compatibles con force-detach. Solo los pods que usan volúmenes/PVCs compatibles con force-detach se eliminan automáticamente.
- La conmutación automática al respaldo y la desconexión forzada se ejecutan dentro del pod trident-controller. Si el nodo que aloja trident-controller falla, la conmutación automática al respaldo se retrasará hasta que K8s mueva el pod a un nodo saludable.

Integración de soluciones personalizadas de comprobación del estado del nodo

Puedes reemplazar Node Healthcheck Operator con herramientas alternativas de detección de fallos de nodo para activar la conmutación automática al respaldo. Para garantizar la compatibilidad con el mecanismo de conmutación automática al respaldo, tu solución personalizada debe:

- Crea un TNR cuando se detecte una falla en un nodo, usando el nombre del nodo fallido como nombre CR del TNR.
- Elimina el TNR cuando el nodo se haya recuperado y el TNR esté en el estado Succeeded.

Seguridad

Seguridad

Usa las recomendaciones que se enumeran aquí para asegurarte de que tu instalación de Trident sea segura.

Ejecuta Trident en su propio espacio de nombres

Es importante evitar que las aplicaciones, los administradores de aplicaciones, los usuarios y las aplicaciones de administración accedan a las definiciones de objetos de Trident o a los pods para garantizar un almacenamiento confiable y bloquear posibles actividades maliciosas.

Para separar las demás aplicaciones y usuarios de Trident, siempre instala Trident en su propio espacio de nombres de Kubernetes (`trident`). Poner Trident en su propio espacio de nombres asegura que solo el personal administrativo de Kubernetes tenga acceso al pod de Trident y a los artefactos (como los secretos de backend y CHAP, si aplica) almacenados en los objetos CRD con espacio de nombres. Debes asegurarte de permitir que solo los administradores tengan acceso al espacio de nombres de Trident y así acceso a la `tridentctl` aplicación.

Usa la autenticación CHAP con backends SAN de ONTAP

Trident admite la autenticación basada en CHAP para cargas de trabajo ONTAP SAN (usando los `ontap-san` y `ontap-san-economy` controladores). NetApp recomienda usar CHAP bidireccional con Trident para la autenticación entre un host y el backend de almacenamiento.

Para los backends de ONTAP que usan los controladores de almacenamiento SAN, Trident puede configurar

CHAP bidireccional y administrar los nombres de usuario y secretos de CHAP a través de `tridentctl`. Consulta "Prepárate para configurar el backend con controladores SAN de ONTAP" para entender cómo Trident configura CHAP en los backends de ONTAP.

Usa la autenticación CHAP con NetApp HCI y SolidFire backends

NetApp recomienda implementar CHAP bidireccional para garantizar la autenticación entre un host y los backends NetApp HCI y SolidFire. Trident utiliza un objeto secreto que incluye dos contraseñas CHAP por inquilino. Cuando se instala Trident, administra los secretos CHAP y los almacena en un objeto CR `tridentvolume` para el PV correspondiente. Cuando creas un PV, Trident utiliza los secretos CHAP para iniciar una sesión iSCSI y comunicarse con el sistema NetApp HCI y SolidFire mediante CHAP.

Los volúmenes que crea Trident no están asociados con ningún Volume Access Group.

Usa Trident con NVE y NAE

NetApp ONTAP proporciona cifrado de datos en reposo para proteger la información confidencial en caso de que un disco sea robado, devuelto o reutilizado. Para más información, consulta "Descripción general de la configuración de NetApp Volume Encryption".

- Si NAE está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NAE habilitado.
  - Puedes configurar el indicador de cifrado NVE en `""` para crear volúmenes habilitados para NAE.
- Si NAE no está habilitado en el backend, cualquier volumen aprovisionado en Trident tendrá NVE habilitado a menos que el indicador de cifrado NVE esté configurado en `false` (el valor predeterminado) en la configuración del backend.

Los volúmenes creados en Trident en un backend habilitado para NAE deben ser cifrados con NVE o NAE.

- Puedes configurar el indicador de cifrado NVE a `true` en la configuración del backend de Trident para anular el cifrado NAE y usar una clave de cifrado específica por volumen.
- Al configurar el indicador de cifrado NVE en `false` un backend con NAE habilitado, se crea un volumen con NAE habilitado. No puedes deshabilitar el cifrado NAE configurando el indicador de cifrado NVE en `false`.

- Puedes crear manualmente un volumen NVE en Trident configurando explícitamente el indicador de cifrado NVE en `true`.

Para obtener más información sobre las opciones de configuración del backend, consulta:

- "Opciones de configuración de ONTAP SAN"
- "Opciones de configuración de ONTAP NAS"

Linux Unified Key Setup (LUKS)

Puedes habilitar Linux Unified Key Setup (LUKS) para cifrar volúmenes ONTAP SAN y ONTAP SAN ECONOMY en Trident. Trident admite la rotación de frases de contraseña y la expansión de volúmenes cifrados con LUKS.

En Trident, los volúmenes cifrados con LUKS utilizan el cifrado y modo aes-xts-plain64, como se recomienda

en "NIST".

> El cifrado LUKS no es compatible con sistemas ASA r2. Para obtener información sobre sistemas ASA r2, consulta "Conoce los sistemas de almacenamiento ASA r2".

Antes de empezar

- Los nodos de trabajo deben tener instalado cryptsetup 2.1 o superior (pero inferior a 3.0). Para más información, visita "Gitlab: cryptsetup".
- Por razones de rendimiento, NetApp recomienda que los nodos de trabajo admitan Advanced Encryption Standard New Instructions (AES-NI). Para verificar la compatibilidad con AES-NI, ejecuta el siguiente comando:

```
grep "aes" /proc/cpuinfo
```

Si no se devuelve nada, tu procesador no es compatible con AES-NI. Para más información sobre AES-NI, visita: "Intel: Instrucciones de cifrado avanzado estándar (AES-NI)".

Habilita el cifrado LUKS

Puedes habilitar el cifrado por volumen del lado del host usando Linux Unified Key Setup (LUKS) para los volúmenes ONTAP SAN y ONTAP SAN ECONOMY.

Pasos

1. Define los atributos de cifrado LUKS en la configuración del backend. Para más información sobre las opciones de configuración del backend para ONTAP SAN, consulta "Opciones de configuración de ONTAP SAN".

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. Usa `parameters.selector` para definir los grupos de almacenamiento usando cifrado LUKS. Por ejemplo:

```
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. Crea un secreto que contenga la contraseña LUKS. Por ejemplo:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitaciones

Los volúmenes cifrados con LUKS no pueden aprovechar la deduplicación y la compresión de ONTAP.

Configuración de backend para importar volúmenes LUKS

Para importar un volumen LUKS, debes establecer `luksEncryption` en (`true` en el backend. La opción `luksEncryption` le dice a Trident si el volumen es compatible con LUKS (`true` o no es compatible con LUKS (`false`, como se muestra en el siguiente ejemplo.

```
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'
```

Configuración de PVC para importar volúmenes LUKS

Para importar volúmenes LUKS de forma dinámica, configura la anotación `trident.netapp.io/luksEncryption` a `true` e incluye una clase de almacenamiento habilitada para LUKS en la PVC como se muestra en este ejemplo.

```
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
```

Rotar una frase de contraseña LUKS

Puedes rotar la frase de contraseña LUKS y confirmar la rotación.

> No olvides una contraseña hasta que hayas verificado que ya no está referenciada por ningún volumen, instantánea o secreto. Si se pierde una contraseña referenciada, puede que no puedas montar el volumen y los datos permanecerán cifrados e inaccesibles.

Acerca de esta tarea

La rotación de la frase de contraseña LUKS ocurre cuando se crea un pod que monta el volumen después de especificar una nueva frase de contraseña LUKS. Cuando se crea un nuevo pod, Trident compara la frase de contraseña LUKS en el volumen con la frase de contraseña activa en el secreto.

- Si la frase de contraseña del volumen no coincide con la frase de contraseña activa en el secreto, ocurre una rotación.
- Si la frase de contraseña del volumen coincide con la frase de contraseña activa en el secreto, el `previous-luks-passphrase` parámetro se ignora.

Pasos

1. Añade los `node-publish-secret-name` y `node-publish-secret-namespace` parámetros StorageClass. Por ejemplo:

```
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. Identifica las frases de contraseña existentes en el volumen o la snapshot.

Volumen

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Actualiza el secreto LUKS del volumen para especificar las frases de contraseña nueva y anterior. Asegúrate de que `previous-luke-passphrase-name` y `previous-luks-passphrase` coincidan con la frase de contraseña anterior.

```
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. Crea un nuevo pod que monte el volumen. Esto es necesario para iniciar la rotación.
5. Verifica que la frase de contraseña haya sido rotada.

Volumen

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Resultados

La frase de contraseña se rotó cuando solo se devuelve la nueva frase de contraseña en el volumen y la instantánea.

Si se devuelven dos contraseñas, por ejemplo `luksPassphraseNames: ["B", "A"]`, la rotación está incompleta. Puedes activar un nuevo pod para intentar completar la rotación.

Habilita la expansión de volumen

Puedes habilitar la expansión de volumen en un volumen cifrado con LUKS.

Pasos

1. Habilita la `CSINodeExpandSecret` puerta de funciones (beta 1.25+). Consulta "Kubernetes 1.25: usa secretos para la expansión de volúmenes CSI basada en nodos" para más detalles.
2. Añade los `node-expand-secret-name` y `node-expand-secret-namespace` parámetros StorageClass. Por ejemplo:

```
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
```

Resultados

Cuando inicias la expansión del almacenamiento en línea, el kubelet pasa las credenciales apropiadas al driver.

Cifrado Kerberos en vuelo

Mediante el cifrado Kerberos en vuelo, puedes mejorar la seguridad de acceso a los datos habilitando el cifrado para el tráfico entre tu clúster gestionado y el backend de almacenamiento.

Trident admite el cifrado Kerberos para ONTAP como backend de almacenamiento:

- **ONTAP local** - Trident admite el cifrado Kerberos sobre conexiones NFSv3 y NFSv4 desde Red Hat OpenShift y clústeres Kubernetes upstream a volúmenes ONTAP locales.

Puedes crear, borrar, redimensionar, crear snapshots, clonar, clonar de solo lectura e importar volúmenes que usan cifrado NFS.

Configura el cifrado Kerberos en vuelo con volúmenes ONTAP locales

Puedes activar el cifrado Kerberos en el tráfico de almacenamiento entre tu clúster gestionado y un backend de almacenamiento ONTAP local.

> El cifrado Kerberos para el tráfico NFS con backends de almacenamiento ONTAP locales solo es compatible usando el controlador de almacenamiento `ontap-nas`.

Antes de empezar

- Asegúrate de que tienes acceso a la utilidad `tridentctl`.
- Asegúrate de que tienes acceso de administrador al backend de almacenamiento de ONTAP.
- Asegúrate de que sabes el nombre del volumen o los volúmenes que vas a compartir desde el backend de almacenamiento de ONTAP.
- Asegúrate de que has preparado la VM de almacenamiento ONTAP para admitir el cifrado Kerberos para volúmenes NFS. Consulta "Habilita Kerberos en un dataLIF" para obtener instrucciones.
- Asegúrate de que cualquier volumen NFSv4 que uses con cifrado Kerberos esté configurado correctamente. Consulta la sección NetApp NFSv4 Domain Configuration (página 13) de "NetApp NFSv4 mejoras y guía de mejores prácticas".

Añade o modifica las políticas de exportación de ONTAP

Necesitas añadir reglas a las políticas de exportación de ONTAP existentes o crear nuevas políticas de exportación que admitan el cifrado Kerberos para el volumen raíz de la máquina virtual de almacenamiento ONTAP, así como para cualquier volumen ONTAP compartido con el clúster de Kubernetes ascendente. Las reglas de las políticas de exportación que añadas, o las nuevas políticas de exportación que crees, deben admitir los siguientes protocolos de acceso y permisos de acceso:

Protocolos de acceso

Configura la política de exportación con los protocolos de acceso NFS, NFSv3 y NFSv4.

Datos de acceso

Puedes configurar una de las tres versiones diferentes de cifrado Kerberos, según lo que necesites para el volumen:

- **Kerberos 5** - (autenticación y cifrado)
- **Kerberos 5i** - (autenticación y cifrado con protección de identidad)
- **Kerberos 5p** - (autenticación y cifrado con protección de identidad y privacidad)

Configura la regla de exportación de ONTAP con los permisos de acceso adecuados. Por ejemplo, si los clusters van a montar los volúmenes NFS con una mezcla de cifrado Kerberos 5i y Kerberos 5p, usa la siguiente configuración de acceso:

Tipo	Acceso de solo lectura	Acceso de lectura/escritura	Acceso de superusuario
UNIX	Habilitado	Habilitado	Habilitado
Kerberos 5i	Habilitado	Habilitado	Habilitado
Kerberos 5p	Habilitado	Habilitado	Habilitado

Consulta la siguiente documentación para saber cómo crear políticas de exportación de ONTAP y reglas de políticas de exportación:

- "Crea una política de exportación"
- "Añade una regla a una política de exportación"

Crea un backend de almacenamiento

Puedes crear una configuración de backend de almacenamiento Trident que incluya la capacidad de cifrado Kerberos.

Acerca de esta tarea

Cuando creas un archivo de configuración de backend de almacenamiento que configura el cifrado Kerberos, puedes especificar una de las tres versiones diferentes de cifrado Kerberos usando el parámetro `spec.nfsMountOptions`:

- `spec.nfsMountOptions: sec=krb5` (autenticación y cifrado)
- `spec.nfsMountOptions: sec=krb5i` (autenticación y cifrado con protección de identidad)
- `spec.nfsMountOptions: sec=krb5p` (autenticación y cifrado con protección de identidad y privacidad)

Especifica solo un nivel de Kerberos. Si especificas más de un nivel de cifrado Kerberos en la lista de parámetros, solo se usa la primera opción.

Pasos

1. En el clúster gestionado, crea un archivo de configuración de backend de almacenamiento usando el siguiente ejemplo. Reemplaza los valores entre corchetes <> con información de tu entorno:

```
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. Usa el archivo de configuración que creaste en el paso anterior para crear el backend:

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

Si la creación del backend falla, algo está mal con la configuración del backend. Puedes ver los logs para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs
```

Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando create.

Crear una clase de almacenamiento

Puedes crear una clase de almacenamiento para aprovisionar volúmenes con cifrado Kerberos.

Acerca de esta tarea

Cuando creas un objeto de clase de almacenamiento, puedes especificar una de las tres versiones diferentes de cifrado Kerberos usando el parámetro `mountOptions`:

- `mountOptions: sec=krb5` (autenticación y cifrado)
- `mountOptions: sec=krb5i` (autenticación y cifrado con protección de identidad)
- `mountOptions: sec=krb5p` (autenticación y cifrado con protección de identidad y privacidad)

Especifica solo un nivel de Kerberos. Si especificas más de un nivel de cifrado Kerberos en la lista de parámetros, solo se usa la primera opción. Si el nivel de cifrado que especificaste en la configuración del backend de almacenamiento es diferente al nivel que especificas en el objeto de clase de almacenamiento, el objeto de clase de almacenamiento tiene prioridad.

Pasos

1. Crea un objeto de Kubernetes StorageClass usando el siguiente ejemplo:

```
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. Crea la clase de almacenamiento:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Asegúrate de que la clase de almacenamiento se haya creado:

```
kubectl get sc ontap-nas-sc
```

   Deberías ver una salida similar a la siguiente:

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provisiona volúmenes

Después de crear un backend de almacenamiento y una clase de almacenamiento, ahora puedes aprovisionar un volumen. Para obtener instrucciones, consulta "Aprovisiona un volumen".

Configura el cifrado Kerberos en tránsito con volúmenes de Azure NetApp Files

Puedes habilitar el cifrado Kerberos en el tráfico de almacenamiento entre tu clúster administrado y un único backend de almacenamiento de Azure NetApp Files o un grupo virtual de backends de almacenamiento de Azure NetApp Files.

Antes de empezar

- Asegúrate de haber habilitado Trident en el clúster Red Hat OpenShift administrado.
- Asegúrate de que tienes acceso a la utilidad `tridentctl`.
- Asegúrate de haber preparado el backend de almacenamiento Azure NetApp Files para el cifrado Kerberos, tomando en cuenta los requisitos y siguiendo las instrucciones en "Documentación de Azure NetApp Files".
- Asegúrate de que cualquier volumen NFSv4 que uses con cifrado Kerberos esté configurado correctamente. Consulta la sección NetApp NFSv4 Domain Configuration (página 13) de "NetApp NFSv4 mejoras y guía de mejores prácticas".

Crea un backend de almacenamiento

Puedes crear una configuración de backend de almacenamiento de Azure NetApp Files que incluya la capacidad de cifrado Kerberos.

Acerca de esta tarea

Cuando creas un archivo de configuración de backend de almacenamiento que configura el cifrado Kerberos, puedes definirlo para que se aplique en uno de dos niveles posibles:

- El **nivel de backend de almacenamiento** usando el campo `spec.kerberos`
- El **nivel de pool virtual** usando el `spec.storage.kerberos` campo

Cuando defines la configuración en el nivel de grupo virtual, el grupo se selecciona usando la etiqueta en la clase de almacenamiento.

En cualquier nivel, puedes especificar una de las tres versiones diferentes del cifrado Kerberos:

- `kerberos: sec=krb5` (autenticación y cifrado)
- `kerberos: sec=krb5i` (autenticación y cifrado con protección de identidad)
- `kerberos: sec=krb5p` (autenticación y cifrado con protección de identidad y privacidad)

Pasos

1. En el clúster administrado, crea un archivo de configuración de backend de almacenamiento usando uno de los siguientes ejemplos, según dónde necesites definir el backend de almacenamiento (a nivel de backend de almacenamiento o a nivel de pool virtual). Reemplaza los valores entre corchetes <> con la información de tu entorno:

Ejemplo de nivel de backend de almacenamiento

```
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
```

Ejemplo de nivel de virtual pool

```
---
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. Usa el archivo de configuración que creaste en el paso anterior para crear el backend:

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

Si la creación del backend falla, algo está mal con la configuración del backend. Puedes ver los logs para determinar la causa ejecutando el siguiente comando:

```
tridentctl logs
```

Después de identificar y corregir el problema con el archivo de configuración, puedes volver a ejecutar el comando create.

Crear una clase de almacenamiento

Puedes crear una clase de almacenamiento para aprovisionar volúmenes con cifrado Kerberos.

Pasos

1. Crea un objeto de Kubernetes StorageClass usando el siguiente ejemplo:

```
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. Crea la clase de almacenamiento:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Asegúrate de que la clase de almacenamiento se haya creado:

```
kubectl get sc -sc-nfs
```

Deberías ver una salida similar a la siguiente:

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Provisiona volúmenes

Después de crear un backend de almacenamiento y una clase de almacenamiento, ahora puedes aprovisionar un volumen. Para obtener instrucciones, consulta "Aprovisiona un volumen".

Protege las aplicaciones con Trident Protect

Conoce Trident Protect

NetApp Trident Protect proporciona capacidades avanzadas de gestión de datos de aplicaciones que mejoran la funcionalidad y la disponibilidad de aplicaciones Kubernetes con estado respaldadas por sistemas de almacenamiento NetApp ONTAP y el aprovisionador de almacenamiento NetApp Trident CSI. Trident Protect simplifica la gestión, protección y movimiento de cargas de trabajo en contenedores en nubes públicas y entornos locales. También ofrece capacidades de automatización a través de su API y CLI.

Puedes proteger las aplicaciones con Trident Protect creando recursos personalizados (CR) o usando la CLI de Trident Protect.

¿Qué sigue?

Puedes informarte sobre los requisitos de Trident Protect antes de instalarlo:

- "Requisitos de Trident Protect"

Instala Trident Protect

Requisitos de Trident Protect

Empieza verificando la preparación de tu entorno operativo, clústeres de aplicaciones, aplicaciones y licencias. Asegúrate de que tu entorno cumple estos requisitos para desplegar y operar Trident Protect.

Compatibilidad de Trident Protect Kubernetes cluster

Trident Protect es compatible con una amplia gama de ofertas de Kubernetes totalmente gestionadas y autogestionadas, incluyendo:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- Kubernetes upstream

- Las copias de seguridad de Trident Protect solo son compatibles con los nodos de cómputo Linux. Los nodos de cómputo Windows no son compatibles para las operaciones de backup.
- Asegúrate de que el clúster en el que instales Trident Protect esté configurado con un controlador de instantáneas en funcionamiento y los CRD relacionados. Para instalar un controlador de instantáneas, consulta "estas instrucciones".
- Asegúrate de que existe al menos un VolumeSnapshotClass. Para más información, consulta "VolumeSnapshotClass".

Compatibilidad del backend de almacenamiento Trident Protect

Trident Protect es compatible con los siguientes storage backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- Matrices de almacenamiento ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Asegúrate de que tu backend de almacenamiento cumple con los siguientes requisitos:

- Asegúrate de que el almacenamiento NetApp conectado al clúster está usando Trident 24.02 o una versión más nueva (se recomienda Trident 24.10).
- Asegúrate de que tienes un backend de almacenamiento NetApp ONTAP.
- Asegúrate de que has configurado un bucket de almacenamiento de objetos para guardar los backups.
- Crea cualquier espacio de nombres de aplicaciones que planees usar para aplicaciones o para operaciones de gestión de datos de aplicaciones. Trident Protect no crea estos espacios de nombres por ti; si especificas un espacio de nombres inexistente en un recurso personalizado, la operación fallará.

Requisitos para los volúmenes de nas-economy

Trident Protect admite operaciones de backup y restauración en volúmenes nas-economy. Las snapshots, los clones y la replicación de SnapMirror en volúmenes nas-economy no son compatibles actualmente. Necesitas habilitar un directorio de snapshots para cada volumen nas-economy que planees usar con Trident Protect.

Algunas aplicaciones no son compatibles con volúmenes que utilizan un directorio de instantáneas. Para estas aplicaciones, necesitas ocultar el directorio de instantáneas ejecutando el siguiente comando en el sistema de almacenamiento ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Puedes activar el directorio de instantáneas ejecutando el siguiente comando para cada volumen nas-economy, reemplazando `<volume-UUID>` por el UUID del volumen que quieres cambiar:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Puedes habilitar los directorios de instantáneas por defecto para los nuevos volúmenes configurando la opción del backend de Trident `snapshotDir` a `true`. Los volúmenes existentes no se ven afectados.

Protegiendo datos con KubeVirt VMs

Trident Protect proporciona capacidades de congelación y descongelación del sistema de archivos para las máquinas virtuales KubeVirt durante las operaciones de protección de datos para garantizar la consistencia de los datos. El método de configuración y el comportamiento predeterminado para las operaciones de congelación de máquinas virtuales varía entre las versiones de Trident Protect, y las versiones más recientes ofrecen una configuración simplificada mediante parámetros del Helm chart.

Durante las operaciones de restauración, cualquier `VirtualMachineSnapshots` creado para una máquina virtual (VM) no se restaura.

Trident Protect 25.10 y versiones más recientes

Trident Protect congela y descongela automáticamente los sistemas de archivos KubeVirt durante las operaciones de protección de datos para garantizar la coherencia. A partir de Trident Protect 25.10, puedes desactivar este comportamiento usando el parámetro `vm.freeze` durante la instalación del chart de Helm. El parámetro está activado por defecto.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 a 25.06

A partir de Trident Protect 24.10.1, Trident Protect congela y descongela automáticamente los sistemas de archivos KubeVirt durante las operaciones de protección de datos. Opcionalmente, puedes desactivar este comportamiento automático usando el siguiente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 no garantiza automáticamente un estado consistente para los sistemas de archivos de KubeVirt VM durante las operaciones de protección de datos. Si quieres proteger tus datos de KubeVirt VM usando Trident Protect 24.10, necesitas habilitar manualmente la funcionalidad de congelar/descongelar para los sistemas de archivos antes de la operación de protección de datos. Esto asegura que los sistemas de archivos estén en un estado consistente.

Puedes configurar Trident Protect 24.10 para gestionar la congelación y descongelación del sistema de archivos de la máquina virtual durante las operaciones de protección de datos mediante "configurando la virtualización" y luego usar el siguiente comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Requisitos para SnapMirror replication

NetApp SnapMirror replication está disponible para su uso con Trident Protect para las siguientes soluciones ONTAP:

- Sistemas NetApp FAS, AFF y ASA en las instalaciones. La replicación SnapMirror con Trident protect no es compatible actualmente para los sistemas ASA r2.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

Requisitos del clúster ONTAP para la replicación de SnapMirror

Asegúrate de que tu clúster ONTAP cumple los siguientes requisitos si planeas usar la replicación SnapMirror:

- **NetApp Trident**: NetApp Trident debe existir en ambos clústeres Kubernetes de origen y destino que utilizan ONTAP como backend. Trident Protect admite la replicación con la tecnología NetApp SnapMirror usando clases de almacenamiento respaldadas por los siguientes controladores:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (requiere ONTAP versión mínima 9.15.1)
- **Licencias**: Las licencias asíncronas de ONTAP SnapMirror usando el paquete Data Protection deben estar habilitadas tanto en el clúster ONTAP de origen como en el clúster de destino. Consulta "Resumen de licencias de SnapMirror en ONTAP" para más información.

  A partir de ONTAP 9.10.1, todas las licencias se entregan como un archivo de licencia NetApp (NLF), que es un único archivo que habilita varias funciones. Consulta "Licencias incluidas con ONTAP One" para más información.

  Solo se admite la protección asíncrona de SnapMirror.

Consideraciones de peering para la replicación SnapMirror

Asegúrate de que tu entorno cumple los siguientes requisitos si planeas usar el peering de backend de almacenamiento:

- **Cluster y SVM**: Los backends de almacenamiento ONTAP deben ser peered. Consulta "Descripción general de clúster y SVM peering" para más información.

  > Asegúrate de que los nombres de SVM usados en la relación de replicación entre dos clústeres ONTAP sean únicos.

- **NetApp Trident y SVM**: Las SVM remotas peered deben estar disponibles para NetApp Trident en el clúster de destino.
- **Backends gestionados**: necesitas agregar y gestionar ONTAP storage backends en Trident Protect para crear una relación de replicación.

Configuración de Trident / ONTAP para SnapMirror replication

Trident Protect requiere que configures al menos un backend de almacenamiento que admita la replicación tanto para el clúster de origen como para el clúster de destino. Si el clúster de origen y el clúster de destino son el mismo, la aplicación de destino debería usar un backend de almacenamiento diferente al de la aplicación de origen para lograr la mejor resiliencia.

Requisitos del clúster de Kubernetes para la replicación de SnapMirror

Asegúrate de que tus clústeres Kubernetes cumplen los siguientes requisitos:

- **Accesibilidad de AppVault**: tanto el clúster de origen como el de destino deben tener acceso a la red para leer y escribir en el AppVault para la replicación de objetos de aplicación.
- **Conectividad de red**: configura las reglas del cortafuegos, los permisos de los buckets y las listas de IP permitidas para permitir la comunicación entre ambos clústeres y el AppVault a través de WANs.

> Muchos entornos empresariales aplican estrictas políticas de cortafuegos en las conexiones WAN. Verifica estos requisitos de red con tu equipo de infraestructura antes de configurar la replicación.

Instala y configura Trident Protect

Si tu entorno cumple los requisitos para Trident Protect, puedes seguir estos pasos para instalar Trident Protect en tu clúster. Puedes obtener Trident Protect de NetApp o instalarlo desde tu propio registro privado. Instalarlo desde un registro privado es útil si tu clúster no puede acceder a Internet.

Instala Trident Protect

Instala Trident Protect de NetApp

Pasos

1. Agrega el repositorio Trident Helm:

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

2. Usa Helm para instalar Trident Protect. Reemplaza `<name-of-cluster>` por un nombre de clúster, que se asignará al clúster y se usará para identificar las copias de seguridad y las instantáneas del clúster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. Opcionalmente, para activar el registro de depuración (recomendado para la solución de problemas), usa:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

   El registro de depuración ayuda a soporte de NetApp a solucionar problemas sin requerir cambios en el nivel de registro ni la reproducción del problema.

Instala Trident Protect desde un registro privado

Puedes instalar Trident Protect desde un registro de imágenes privado si tu clúster de Kubernetes no puede acceder a Internet. En estos ejemplos, reemplaza los valores entre corchetes por información de tu entorno:

Pasos

1. Descarga las siguientes imágenes en tu máquina local, actualiza las etiquetas y luego súbelas a tu registro privado:

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Por ejemplo:

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Para obtener el gráfico Helm, primero descarga el gráfico Helm en una máquina con acceso a internet usando `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`, luego copia el archivo `trident-protect-100.2510.0.tgz` resultante a tu entorno sin conexión e instala usando `helm install trident-protect ./trident-protect-100.2510.0.tgz` en vez de la referencia del repositorio en el paso final.

2. Crea el espacio de nombres del sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Inicia sesión en el registro:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Crea un pull secret para usarlo en la autenticación del registro privado:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Agrega el repositorio Trident Helm:

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

6. Crea un archivo llamado `protectValues.yaml`. Asegúrate de que contiene la siguiente configuración de Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> Los valores `imageRegistry` y `imagePullSecrets` se aplican a todas las imágenes de componentes, incluyendo `resourcebackup` y `resourcerestore`. Si envías imágenes a una ruta de repositorio específica dentro de tu registro (por ejemplo, `example.com:443/my-repo`), incluye la ruta completa en el campo de registro. Esto asegurará que todas las imágenes se extraigan de `<private-registry-url>/<image-name>:<tag>`.

7. Usa Helm para instalar Trident Protect. Reemplaza `<name_of_cluster>` por un nombre de clúster, que se asignará al clúster y se usará para identificar los backups y las snapshots del clúster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Opcionalmente, para activar el registro de depuración (recomendado para la solución de problemas), usa:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

El registro de depuración ayuda a soporte de NetApp a solucionar problemas sin requerir cambios en el nivel de registro ni la reproducción del problema.

Para más opciones de configuración del gráfico de Helm, incluidos los ajustes de AutoSupport y el filtrado de espacios de nombres, consulta "Personaliza la instalación de Trident Protect".

Instala el plugin CLI de Trident Protect

Puedes usar el complemento de línea de comandos de Trident Protect, que es una extensión de la utilidad Trident `tridentctl` para crear e interactuar con los recursos personalizados (CRs) de Trident Protect.

Instala el plugin CLI de Trident Protect

Antes de usar la utilidad de línea de comandos, necesitas instalarla en la máquina que usas para acceder a tu clúster. Sigue estos pasos, dependiendo de si tu máquina usa una CPU x64 o ARM.

Descarga el plugin para Linux AMD64 CPUs

Pasos

1. Descarga el plugin CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

Descarga el plugin para CPUs Linux ARM64

Pasos

1. Descarga el plugin CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

Descarga el plugin para Mac con CPU AMD64

Pasos

1. Descarga el plugin CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

Descarga el plugin para Mac ARM64 CPUs

Pasos

1. Descarga el plugin CLI de Trident Protect:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. Habilita los permisos de ejecución para el binario del plugin:

```
chmod +x tridentctl-protect
```

2. Copia el archivo binario del complemento en una ubicación que esté definida en tu variable PATH. Por ejemplo, `/usr/bin` o `/usr/local/bin` (puede que necesites privilegios elevados):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Opcionalmente, puedes copiar el binario del plugin en una ubicación de tu directorio inicial. En este caso, se recomienda asegurarte de que la ubicación sea parte de tu variable PATH:

```
cp ./tridentctl-protect ~/bin/
```

Copiar el complemento en una ubicación en tu variable PATH te permite usar el complemento escribiendo `tridentctl-protect` o `tridentctl protect` desde cualquier ubicación.

Ver la ayuda del plugin CLI de Trident

Puedes usar las funciones de ayuda integradas en el plugin para obtener ayuda detallada sobre las capacidades del plugin:

Pasos

1. Utiliza la función de ayuda para ver la guía de uso:

```
tridentctl-protect help
```

Habilitar el autocompletado de comandos

Después de que hayas instalado el complemento CLI de Trident Protect, puedes habilitar el autocompletado para ciertos comandos.

Habilita el autocompletado para la shell Bash

Pasos

1. Crea el script de finalización:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Crea un nuevo directorio en tu directorio inicial para contener el script:

```
mkdir -p ~/.bash/completions
```

3. Mueve el script descargado al directorio `~/.bash/completions`:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Agrega la siguiente línea al archivo `~/.bashrc` en tu directorio inicial:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Habilita el autocompletado para la Z shell

Pasos

1. Crea el script de finalización:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Crea un nuevo directorio en tu directorio inicial para contener el script:

```
mkdir -p ~/.zsh/completions
```

3. Mueve el script descargado al directorio `~/.zsh/completions`:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Agrega la siguiente línea al archivo `~/.zprofile` en tu directorio inicial:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Resultado

En tu próximo inicio de sesión en el shell, puedes usar el autocompletado de comandos con el plugin tridentctl-protect.

Personaliza la instalación de Trident Protect

Puedes personalizar la configuración predeterminada de Trident Protect para satisfacer los requisitos específicos de tu entorno.

Especifica los límites de recursos del contenedor Trident Protect

Puedes usar un archivo de configuración para especificar los límites de recursos para los contenedores de Trident Protect después de instalar Trident Protect. Configurar los límites de recursos te permite controlar cuántos recursos del clúster consumen las operaciones de Trident Protect.

Pasos

1. Crea un archivo llamado `resourceLimits.yaml`.
2. Rellena el archivo con las opciones de límite de recursos para los contenedores Trident Protect según las necesidades de tu entorno.

   El siguiente archivo de configuración de ejemplo muestra los ajustes disponibles y contiene los valores predeterminados para cada límite de recursos:

```
---
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. Aplica los valores del archivo `resourceLimits.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personaliza las restricciones del contexto de seguridad

Puedes usar un archivo de configuración para modificar las security context constraint (SCCs) de OpenShift para los contenedores de Trident Protect después de instalar Trident Protect. Estas constraints definen restricciones de seguridad para los pods en un clúster de Red Hat OpenShift.

Pasos

1. Crea un archivo llamado `sccconfig.yaml`.
2. Agrega la opción SCC al archivo y modifica los parámetros según las necesidades de tu entorno.

   El siguiente ejemplo muestra los valores predeterminados de los parámetros para la opción SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Esta tabla describe los parámetros para la opción SCC:

Parámetro	Descripción	Predeterminado
crear	Determina si se puede crear un recurso SCC. Se creará un recurso SCC solo si `scc.create` está configurado en `true` y el proceso de instalación de Helm identifica un entorno OpenShift. Si no se está operando en OpenShift o si `scc.create` está configurado en `false`, no se creará ningún recurso SCC.	verdadero
nombre	Especifica el nombre del SCC.	trident-protect-job
prioridad	Define la prioridad del SCC. Los SCC con valores de prioridad más altos se evalúan antes que los que tienen valores más bajos.	1

3. Aplica los valores del archivo `sccconfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Esto sustituirá los valores por defecto por los especificados en el `sccconfig.yaml` archivo.

Configura los ajustes adicionales del helm chart de Trident Protect

Puedes personalizar la configuración de AutoSupport y el filtrado de espacios de nombres para satisfacer tus requisitos específicos. La siguiente tabla describe los parámetros de configuración disponibles:

Parámetro	Tipo	Descripción
autoSupport.proxy	cadena	Configura una URL de proxy para las conexiones de NetApp AutoSupport. Usa esto para enrutar las cargas de paquetes de soporte a través de un servidor proxy. Ejemplo: `http://my.proxy.url`.
autoSupport.insecure	booleano	Omite la verificación TLS para las conexiones proxy de AutoSupport cuando se establece en `true`. Úsalo solo para conexiones proxy no seguras. (predeterminado: `false`)

Parámetro	Tipo	Descripción
autoSupport.enabled	booleano	Habilita o deshabilita las cargas diarias de paquetes de Trident Protect AutoSupport. Cuando se establece en `false`, las cargas diarias programadas se deshabilitan, pero aún puedes generar manualmente paquetes de soporte. (predeterminado: `true`)
restoreSkipNamespaceAnnotations	cadena	Lista separada por comas de anotaciones de espacios de nombres que se excluirán de las operaciones de copia de seguridad y restauración. Te permite filtrar los espacios de nombres según las anotaciones.
restoreSkipNamespaceLabels	cadena	Lista separada por comas de etiquetas de espacios de nombres que se excluirán de las operaciones de copia de seguridad y restauración. Te permite filtrar los espacios de nombres según las etiquetas.

Puedes configurar estas opciones usando un archivo de configuración YAML o indicadores de línea de comandos:

Usa el archivo YAML

Pasos

1. Crea un archivo de configuración y ponle el nombre `values.yaml`.
2. En el archivo que creaste, agrega las opciones de configuración que quieras personalizar.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Después de rellenar el archivo `values.yaml` con los valores correctos, aplica el archivo de configuración:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Usa el indicador CLI

Pasos

1. Usa el siguiente comando con la bandera `--set` para especificar parámetros individuales:

```
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
```

Restringe los pods de Trident Protect a nodos específicos

Puedes usar la restricción de selección de nodos de Kubernetes nodeSelector para controlar cuáles de tus nodos pueden ejecutar pods de Trident Protect, según las etiquetas de los nodos. Por defecto, Trident Protect está restringido a los nodos que ejecutan Linux. Puedes personalizar aún más estas restricciones según lo que necesites.

Pasos

1. Crea un archivo llamado `nodeSelectorConfig.yaml`.
2. Agrega la opción nodeSelector al archivo y modifica el archivo para añadir o cambiar las etiquetas de nodo

y restringir según las necesidades de tu entorno. Por ejemplo, el siguiente archivo contiene la restricción de SO por defecto, pero también apunta a una región y un nombre de app específicos:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Aplica los valores del archivo `nodeSelectorConfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Esto reemplaza las restricciones predeterminadas por las que especificaste en el archivo `nodeSelectorConfig.yaml`.

Administra Trident Protect

Administra la autorización y el control de acceso de Trident Protect

Trident Protect utiliza el modelo de Kubernetes de control de acceso basado en roles (RBAC). Por defecto, Trident Protect proporciona un único espacio de nombres del sistema y su cuenta de servicio predeterminada asociada. Si tienes una organización con muchos usuarios o necesidades de seguridad específicas, puedes usar las funciones de RBAC de Trident Protect para tener un control más granular sobre el acceso a los recursos y espacios de nombres.

El administrador del clúster siempre tiene acceso a los recursos en el espacio de nombres predeterminado `trident-protect` y también puede acceder a los recursos en todos los demás espacios de nombres. Para controlar el acceso a los recursos y aplicaciones, necesitas crear espacios de nombres adicionales y agregar recursos y aplicaciones a esos espacios de nombres.

Ten en cuenta que ningún usuario puede crear CR de gestión de datos de aplicaciones en el espacio de nombres predeterminado `trident-protect`. Necesitas crear CR de gestión de datos de aplicaciones en un espacio de nombres de aplicaciones (como mejor práctica, crea los CR de gestión de datos de aplicaciones en el mismo espacio de nombres que su aplicación asociada).

Solo los administradores deben tener acceso a los objetos de recursos personalizados privilegiados de Trident Protect, que incluyen:

- **AppVault**: requiere datos de credenciales de cubo
- **AutoSupportBundle**: recoge métricas, registros y otros datos confidenciales de Trident Protect
- **AutoSupportBundleSchedule**: gestiona los calendarios de recogida de registros

Como mejor práctica, usa RBAC para restringir el acceso a objetos privilegiados a los administradores.

Para obtener más información sobre cómo RBAC regula el acceso a los recursos y espacios de nombres, consulta la "Documentación de Kubernetes RBAC".

Para más información sobre las cuentas de servicio, consulta el "Documentación de cuentas de servicio de Kubernetes".

Ejemplo: gestionar el acceso de dos grupos de usuarios

Por ejemplo, una organización tiene un administrador de clúster, un grupo de usuarios de ingeniería y un grupo de usuarios de marketing. El administrador de clúster realizaría las siguientes tareas para crear un entorno donde el grupo de ingeniería y el grupo de marketing tengan acceso solo a los recursos asignados a sus respectivos espacios de nombres.

Paso 1: crea un espacio de nombres para contener los recursos de cada grupo

Crear un espacio de nombres te permite separar lógicamente los recursos y controlar mejor quién tiene acceso a esos recursos.

Pasos

1. Crea un espacio de nombres para el grupo de ingeniería:

```
kubectl create ns engineering-ns
```

2. Crea un espacio de nombres para el grupo de marketing:

```
kubectl create ns marketing-ns
```

Paso 2: crea nuevas cuentas de servicio para interactuar con recursos en cada espacio de nombres

Cada nuevo espacio de nombres que crees viene con una cuenta de servicio por defecto, pero deberías crear una cuenta de servicio para cada grupo de usuarios para que puedas dividir aún más los privilegios entre grupos en el futuro si es necesario.

Pasos

1. Crea una cuenta de servicio para el grupo de ingeniería:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Crea una cuenta de servicio para el grupo de marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Paso 3: crea un secreto para cada nueva cuenta de servicio

Un secreto de cuenta de servicio se usa para autenticarse con la cuenta de servicio y se puede borrar y recrear fácilmente si se ve comprometido.

Pasos

1. Crea un secreto para la cuenta de servicio de ingeniería:

```
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. Crea un secreto para la cuenta del servicio de marketing:

```
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
```

Paso 4: crea un objeto RoleBinding para vincular el objeto ClusterRole a cada nueva cuenta de servicio

Se crea un objeto ClusterRole por defecto cuando instalas Trident Protect. Puedes vincular este ClusterRole a la cuenta de servicio creando y aplicando un objeto RoleBinding.

Pasos

1. Vincula el ClusterRole a la cuenta de servicio de ingeniería:

```
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. Vincula el ClusterRole a la cuenta de servicio de marketing:

```
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
```

Paso 5: probar los permisos

Comprueba que los permisos son correctos.

Pasos

1. Confirma que los usuarios de ingeniería pueden acceder a los recursos de ingeniería:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confirma que los usuarios de ingeniería no pueden acceder a los recursos de marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Paso 6: concede acceso a los objetos de AppVault

Para realizar tareas de gestión de datos como copias de seguridad e instantáneas, el administrador del clúster debe conceder acceso a AppVault objetos a usuarios individuales.

Pasos

1. Crea y aplica un archivo YAML de combinación de AppVault y secreto que le da a un usuario acceso a un AppVault. Por ejemplo, el siguiente CR da acceso a un AppVault al usuario `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. Crea y aplica un Role CR para permitir que los administradores de clúster otorguen acceso a recursos específicos en un namespace. Por ejemplo:

```
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. Crea y aplica una CR RoleBinding para vincular los permisos al usuario eng-user. Por ejemplo:

```
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. Verifica que los permisos sean correctos.

   a. Intenta recuperar la información del objeto AppVault para todos los espacios de nombres:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Deberías ver una salida similar a la siguiente:

```
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. Prueba para ver si el usuario puede obtener la información de AppVault que ahora tiene permiso para acceder:

```
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
```

Deberías ver una salida similar a la siguiente:

```
yes
```

Resultado

Los usuarios a los que has concedido permisos de AppVault deberían poder usar los objetos autorizados de AppVault para operaciones de gestión de datos de la aplicación, y no deberían poder acceder a ningún recurso fuera de los espacios de nombres asignados ni crear nuevos recursos a los que no tengan acceso.

Supervisa los recursos de Trident Protect

Puedes usar las herramientas de código abierto kube-state-metrics, Prometheus y Alertmanager para monitorear la salud de los recursos protegidos por Trident Protect.

El servicio kube-state-metrics genera métricas a partir de la comunicación de la API de Kubernetes. Si lo usas con Trident Protect, te muestra información útil sobre el estado de los recursos en tu entorno.

Prometheus es un conjunto de herramientas que puede ingerir los datos generados por kube-state-metrics y presentarlos como información fácilmente legible sobre estos objetos. Juntos, kube-state-metrics y Prometheus te ofrecen una forma de monitorear la salud y el estado de los recursos que estás gestionando con Trident Protect.

Alertmanager es un servicio que ingiere las alertas enviadas por herramientas como Prometheus y las dirige a los destinos que tú configures.

> Las configuraciones y orientaciones incluidas en estos pasos son solo ejemplos; necesitas personalizarlas para que coincidan con tu entorno. Consulta la siguiente documentación oficial para instrucciones y soporte específicos:
>
> - "documentación de kube-state-metrics"
> - "Documentación de Prometheus"
> - "Documentación de Alertmanager"

Paso 1: instala las herramientas de supervisión

Para habilitar la supervisión de recursos en Trident Protect, necesitas instalar y configurar kube-state-metrics, Prometheus y Alertmanager.

Instala kube-state-metrics

Puedes instalar kube-state-metrics usando Helm.

Pasos

1. Agrega el Helm chart de kube-state-metrics. Por ejemplo:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Aplica el CRD de Prometheus ServiceMonitor al clúster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Crea un archivo de configuración para el Helm chart (por ejemplo, `metrics-config.yaml`). Puedes personalizar el siguiente ejemplo de configuración para que se ajuste a tu entorno:

metrics-config.yaml: configuración del Helm chart de kube-state-metrics

```
---
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. Instala kube-state-metrics desplegando el chart de Helm. Por ejemplo:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configura kube-state-metrics para generar métricas para los recursos personalizados usados por Trident Protect siguiendo las instrucciones en el "documentación de recursos personalizados de kube-state-metrics".

Instala Prometheus

Puedes instalar Prometheus siguiendo las instrucciones en el "Documentación de Prometheus".

Instala Alertmanager

Puedes instalar Alertmanager siguiendo las instrucciones en "Documentación de Alertmanager".

Paso 2: configura las herramientas de supervisión para que funcionen juntas

Después de instalar las herramientas de supervisión, necesitas configurarlas para que funcionen juntas.

Pasos

1. Integra kube-state-metrics con Prometheus. Edita el archivo de configuración de Prometheus (`prometheus.yaml` y añade la información del servicio kube-state-metrics. Por ejemplo:

   prometheus.yaml: integración del servicio kube-state-metrics con 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. Configura Prometheus para enrutar las alertas a Alertmanager. Edita el archivo de configuración de Prometheus (`prometheus.yaml` y agrega la siguiente sección:

prometheus.yaml: enviar alertas a Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Resultado

Prometheus puede ahora recopilar métricas de kube-state-metrics y puede enviar alertas a Alertmanager. Ahora estás listo para configurar qué condiciones activan una alerta y a dónde se deben enviar las alertas.

Paso 3: Configura las alertas y los destinos de alertas

Después de configurar las herramientas para que funcionen juntas, necesitas configurar qué tipo de información activa las alertas y a dónde se deben enviar.

Ejemplo de alerta: fallo de backup

El siguiente ejemplo define una alerta crítica que se activa cuando el estado del recurso personalizado de copia de seguridad se establece en `Error` durante 5 segundos o más. Puedes personalizar este ejemplo para adaptarlo a tu entorno e incluir este fragmento YAML en tu `prometheus.yaml` archivo de configuración:

rules.yaml: define una alerta Prometheus para copias de seguridad fallidas

```
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."
```

Configura Alertmanager para enviar alertas a otros canales

Puedes configurar Alertmanager para enviar notificaciones a otros canales, como correo electrónico, PagerDuty, Microsoft Teams u otros servicios de notificación especificando la configuración respectiva en el archivo `alertmanager.yaml`.

El siguiente ejemplo configura Alertmanager para enviar notificaciones a un canal de Slack. Para adaptar este ejemplo a tu entorno, reemplaza el valor de la clave `api_url` con la URL del webhook de Slack que usas en tu entorno:

alertmanager.yaml: enviar alertas a un canal de Slack

```
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
```

Genera un paquete de soporte Trident Protect

Trident Protect permite a los administradores generar paquetes que incluyen información útil para NetApp Support, incluyendo registros, métricas e información de topología sobre los clústeres y las apps bajo gestión. Si estás conectado a internet, puedes subir paquetes de soporte al NetApp Support Site (NSS) usando un archivo custom resource (CR).

Crea un paquete de soporte usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-support-bundle.yaml`).
2. Configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.triggerType**: (*Obligatorio*) determina si el paquete de soporte se genera de inmediato o se programa. La generación programada del paquete ocurre a las 12AM UTC. Valores posibles:
     - Programado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla si el paquete de soporte debe cargarse en el NetApp Support Site después de generarse. Si no se especifica, el valor predeterminado es `false`. Valores posibles:
     - verdadero
     - false (predeterminado)
   - **spec.dataWindowStart**: (*Opcional*) Una cadena de fecha en formato RFC 3339 que especifica la fecha y hora en que debe comenzar la ventana de datos incluidos en el soporte bundle. Si no se especifica, el valor predeterminado es hace 24 horas. La fecha de ventana más temprana que puedes especificar es hace 7 días.

     Ejemplo de 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. Después de rellenar el archivo `trident-protect-support-bundle.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

Crea un paquete de soporte usando la CLI

Pasos

1. Crea el paquete de soporte, reemplazando los valores entre corchetes por información de tu entorno.

El `trigger-type` determina si el paquete se crea inmediatamente o si la hora de creación la dicta la programación, y puede ser `Manual` o `Scheduled`. La configuración predeterminada es `Manual`.

Por ejemplo:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Supervisa y recupera el paquete de soporte

Después de crear un paquete de soporte usando cualquiera de los dos métodos, puedes supervisar el progreso de su generación y recuperarlo en tu sistema local.

Pasos

1. Espera a que `status.generationState` llegue al estado `Completed`. Puedes monitorear el progreso de la generación con el siguiente comando:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

2. Recupera el paquete de soporte en tu sistema local. Obtén el comando de copia del paquete completo AutoSupport:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Busca el comando `kubectl cp` en la salida y ejecútalo, reemplazando el argumento de destino por tu directorio local preferido.

Actualizar Trident Protect

Puedes actualizar Trident Protect a la última versión para aprovechar las nuevas funciones o correcciones de errores.

- Al actualizar desde la versión 24.10, las instantáneas que se estén ejecutando durante la actualización pueden fallar. Este fallo no impide que se creen futuras instantáneas, ya sean manuales o programadas. Si una instantánea falla durante la actualización, puedes crear manualmente una nueva instantánea para asegurarte de que tu aplicación esté protegida.

  Para evitar posibles fallos, puedes desactivar todas las programaciones de instantáneas antes de la actualización y volver a activarlas después. Sin embargo, esto provoca que se pierdan las instantáneas programadas durante el periodo de actualización.

- Para instalaciones de registro privado, asegúrate de que el Helm chart requerido y las imágenes para la versión de destino estén disponibles en tu registro privado, y verifica que tus valores personalizados de Helm sean compatibles con la nueva versión del chart. Para más información, consulta "Instala Trident Protect desde un registro privado".

Para actualizar Trident Protect, realiza los siguientes pasos.

Pasos

1. Actualiza el repositorio de Trident Helm:

```
helm repo update
```

2. Actualiza los CRD de Trident Protect:

   Este paso es necesario si estás actualizando desde una versión anterior a la 25.06, ya que los CRDs ahora están incluidos en el Helm chart de Trident Protect.

   a. Ejecuta este comando para cambiar la gestión de los CRD de `trident-protect-crds` a `trident-protect`:

```
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. Ejecuta este comando para borrar el secreto de Helm del chart `trident-protect-crds`:

   No desinstales el chart `trident-protect-crds` usando Helm, ya que esto podría eliminar tus CRDs y cualquier dato relacionado.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Actualiza Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

Puedes configurar el nivel de registro durante la actualización añadiendo `--set logLevel=debug` al comando de actualización. El nivel de registro predeterminado es `warn`. Se recomienda el registro de depuración para la solución de problemas, ya que ayuda al soporte de NetApp a diagnosticar problemas sin necesidad de cambiar el nivel de registro ni reproducir el problema.

Gestiona y protege las aplicaciones

Usa los objetos de Trident Protect AppVault para gestionar buckets

El recurso personalizado (CR) del bucket para Trident Protect se conoce como un AppVault. Los objetos AppVault son la representación declarativa del flujo de trabajo de Kubernetes de un bucket de almacenamiento. Un CR AppVault contiene las configuraciones necesarias para que un bucket se use en operaciones de protección, como copias de seguridad, instantáneas, operaciones de restauración y replicación de SnapMirror. Solo los administradores pueden crear AppVaults.

Necesitas crear una CR de AppVault manualmente o desde la línea de comandos cuando realizas operaciones de protección de datos en una aplicación. La CR de AppVault es específica de tu entorno y puedes usar los ejemplos en esta página como guía cuando crees CR de AppVault.

Asegúrate de que el CR AppVault está en el clúster donde está instalado Trident Protect. Si el CR AppVault no existe o no puedes acceder a él, la línea de comandos muestra un error.

Configura la autenticación y las contraseñas de AppVault

Antes de crear un AppVault CR, asegúrate de que el AppVault y el data mover que elijas puedan autenticarse con el proveedor y cualquier recurso relacionado.

Contraseñas del repositorio de data mover

Cuando creas objetos AppVault usando CRs o el complemento de línea de comandos (CLI) de Trident Protect, puedes especificar un secreto de Kubernetes con contraseñas personalizadas para el cifrado de Restic y Kopia. Si no especificas un secreto, Trident Protect usa una contraseña predeterminada.

- Al crear manualmente AppVault CRs, usa el campo **spec.dataMoverPasswordSecretRef** para especificar el secreto.
- Al crear objetos AppVault usando la CLI de Trident Protect, usa el argumento `--data-mover-password-secret-ref` para especificar el secreto.

Crea una contraseña secreta para el repositorio de data mover

Usa los siguientes ejemplos para crear el secreto de la contraseña. Cuando crees objetos AppVault, puedes indicarle a Trident Protect que use este secreto para autenticarse con el repositorio de data mover.

- En función del motor de datos que utilices, solo tienes que incluir la contraseña correspondiente a ese motor de datos. Por ejemplo, si usas Restic y no planeas usar Kopia en el futuro, puedes incluir solo la contraseña de Restic cuando crees el secreto.
- Guarda la contraseña en un lugar seguro. La vas a necesitar para restaurar datos en el mismo clúster o en uno diferente. Si se elimina el clúster o el `trident-protect` namespace, no podrás restaurar tus backups o snapshots sin la contraseña.

Usa un 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
```

Usa la CLI

```
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
```

Permisos IAM de almacenamiento compatible con S3

Cuando accedes a almacenamiento compatible con S3, como Amazon S3, Generic S3, "StorageGrid S3" o "ONTAP S3" usando Trident Protect, necesitas asegurarte de que las credenciales de usuario que proporciones tengan los permisos necesarios para acceder al bucket. El siguiente es un ejemplo de una política que otorga los permisos mínimos requeridos para el acceso con Trident Protect. Puedes aplicar esta política al usuario que administra las políticas de buckets compatibles con S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Para obtener más información sobre las políticas de Amazon S3, consulta los ejemplos en "Documentación de Amazon S3".

EKS Pod Identity para Amazon S3 (AWS) authentication

Trident Protect es compatible con EKS Pod Identity para operaciones de traslado de datos de Kopia. Esta función permite el acceso seguro a los buckets de S3 sin almacenar credenciales de AWS en los secretos de Kubernetes.

Requisitos para EKS Pod Identity con Trident Protect

Antes de usar EKS Pod Identity con Trident Protect, asegúrate de lo siguiente:

- Tu clúster EKS tiene activado Pod Identity.
- Has creado una función de IAM con los permisos necesarios para el bucket de S3. Para saber más, consulta "Permisos IAM de almacenamiento compatible con S3".
- La función IAM está asociada con las siguientes cuentas de servicio de Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Para obtener instrucciones detalladas sobre cómo habilitar Pod Identity y asociar roles IAM con cuentas de servicio, consulta la "Documentación de AWS EKS Pod Identity".

Configuración de AppVault Cuando uses EKS Pod Identity, configura tu CR de AppVault con la `useIAM: true` flag en vez de credenciales explícitas:

```
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 ejemplos de generación de claves para proveedores de nube

Al definir un CR de AppVault, necesitas incluir credenciales para acceder a los recursos alojados por el proveedor, a menos que estés usando autenticación IAM. Cómo generes las claves para las credenciales dependerá del proveedor. A continuación tienes ejemplos de generación de claves por línea de comandos para varios proveedores. Puedes usar los siguientes ejemplos para crear claves para las credenciales de cada proveedor de nube.

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
```

S3 genérico

```
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
```

Ejemplos de creación de AppVault

Los siguientes son ejemplos de definiciones de AppVault para cada proveedor.

AppVault ejemplos de CR

Puedes usar los siguientes ejemplos de CR para crear objetos AppVault para cada proveedor de nube.

- Puedes especificar opcionalmente un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado de los repositorios Restic y Kopia. Consulta Contraseñas del repositorio de data mover para más información.
- Para los objetos de Amazon S3 (AWS) AppVault, puedes especificar opcionalmente un sessionToken, que es útil si usas el inicio de sesión único (SSO) para la autenticación. Este token se crea cuando generas claves para el proveedor en AppVault ejemplos de generación de claves para proveedores de nube.
- Para los objetos S3 AppVault, puedes especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente usando la clave `spec.providerConfig.S3.proxyURL`.

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)

```yaml
---
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
```

Para entornos EKS que usan Pod Identity con Kopia data mover, puedes eliminar la sección `providerCredentials` y agregar `useIAM: true` dentro de la configuración `s3` en su lugar.

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
```

S3 genérico

```
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
```

Ejemplos de creación de AppVault usando la CLI de Trident Protect

Puedes usar los siguientes ejemplos de comandos de la CLI para crear AppVault CRs para cada proveedor.

- Puedes especificar opcionalmente un secreto de Kubernetes que contenga contraseñas personalizadas para el cifrado de los repositorios Restic y Kopia. Consulta Contraseñas del repositorio de data mover para más información.
- Para los objetos S3 AppVault, puedes especificar opcionalmente una URL de proxy de salida para el tráfico S3 saliente usando el argumento `--proxy-url <ip_address:port>`.

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
```

S3 genérico

```
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
```

Opciones de configuración compatibles `providerConfig.s3`

Consulta la siguiente tabla para ver las opciones de configuración del proveedor S3:

Parámetro	Descripción	Predeterminado	Ejemplo
`providerConfig.s3.skipCertValidation`	Desactiva la verificación de certificados SSL/TLS.	false	"true", "false"
`providerConfig.s3.secure`	Habilita la comunicación HTTPS segura con el endpoint S3.	verdadero	"true", "false"
`providerConfig.s3.proxyURL`	Especifica la URL del servidor proxy usado para conectarte a S3.	Ninguno	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Proporciona un certificado CA raíz personalizado para la verificación SSL/TLS.	Ninguno	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Habilita la autenticación IAM para acceder a los buckets S3. Aplicable para EKS Pod Identity.	false	verdadero, falso

Ver información de AppVault

Puedes usar el complemento CLI de Trident Protect para ver información sobre los objetos AppVault que has creado en el clúster.

Pasos

1. Ver el contenido de un AppVault objeto:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Ejemplo de salida:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   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. Opcionalmente, para ver el AppVaultPath para cada recurso, usa la bandera `--show-paths`.

   El nombre del clúster en la primera columna de la tabla solo está disponible si se especificó un nombre de clúster en la instalación de Trident Protect helm. Por ejemplo: `--set clusterName=production1`.

Eliminar un AppVault

Puedes eliminar un objeto AppVault en cualquier momento.

No elimines la clave `finalizers` en el CR AppVault antes de borrar el objeto AppVault. Si lo haces, pueden quedar datos residuales en el bucket AppVault y recursos huérfanos en el clúster.

Antes de empezar

Asegúrate de que has eliminado todos los CR de instantáneas y copias de seguridad que está usando el AppVault que quieres eliminar.

Eliminar un AppVault usando la CLI de Kubernetes

1. Elimina el objeto AppVault, reemplazando `appvault-name` por el nombre del objeto AppVault que quieres eliminar:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Elimina un AppVault usando la CLI de Trident Protect

1. Elimina el objeto AppVault, reemplazando `appvault-name` por el nombre del objeto AppVault que quieres eliminar:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Define una aplicación para gestión con Trident Protect

Puedes definir una aplicación que quieras gestionar con Trident Protect creando una CR de aplicación y una CR asociada AppVault.

Crea un CR de AppVault

Necesitas crear un CR de AppVault que se usará cuando realices operaciones de protección de datos en la aplicación, y el CR de AppVault debe estar en el clúster donde está instalado Trident Protect. El CR de AppVault es específico para tu entorno; para ejemplos de CR de AppVault, consulta "AppVault recursos personalizados."

Define una aplicación

Necesitas definir cada aplicación que quieras gestionar con Trident Protect. Puedes definir una aplicación para gestionarla creando manualmente un CR de aplicación o usando la CLI de Trident Protect.

Agrega una aplicación usando un CR

Pasos

1. Crea el archivo CR de la aplicación de destino:
   a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `maria-app.yaml`).
   b. Configura los siguientes atributos:
      - **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de la aplicación. Ten en cuenta el nombre que elijas porque otros archivos CR necesarios para las operaciones de protección hacen referencia a este valor.
      - **spec.includedNamespaces**: (*Requerido*) Usa el selector de espacio de nombres y de etiqueta para especificar los espacios de nombres y recursos que utiliza la aplicación. El espacio de nombres de la aplicación debe formar parte de esta lista. El selector de etiqueta es opcional y puedes usarlo para filtrar recursos dentro de cada espacio de nombres especificado.
      - **spec.includedClusterScopedResources**: (*Opcional*) Usa este atributo para especificar los recursos de ámbito clúster que se incluirán en la definición de la aplicación. Este atributo te permite seleccionar estos recursos según su grupo, versión, tipo y etiquetas.
        - **groupVersionKind**: (*Obligatorio*) especifica el grupo de API, la versión y el tipo del recurso con alcance de clúster.
        - **labelSelector**: (*Opcional*) Filtra los recursos con ámbito de clúster según sus etiquetas.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotación solo es aplicable a aplicaciones definidas a partir de máquinas virtuales, como en entornos KubeVirt, donde se producen congelaciones del sistema de archivos antes de las instantáneas. Especifica si esta aplicación puede escribir en el sistema de archivos durante una instantánea. Si se establece en true, la aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una instantánea. Si se establece en false, la aplicación ignora la configuración global y el sistema de archivos se congela durante una instantánea. Si se especifica pero la aplicación no tiene máquinas virtuales en la definición de la aplicación, se ignora la anotación. Si no se especifica, la aplicación sigue la "ajuste global de congelación de Trident Protect".

Si necesitas aplicar esta anotación después de que ya se haya creado una aplicación, puedes usar el siguiente comando:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Ejemplo de 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. (*Opcional*) agrega filtrado que incluya o excluya recursos marcados con etiquetas específicas:

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:

     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro de cada elemento (group, kind, version) coinciden como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
       - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
       - **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.

- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

> Cuando se utilizan tanto `resourceFilter` como `labelSelector`, `resourceFilter` se ejecuta primero y luego `labelSelector` se aplica a los recursos resultantes.

Por ejemplo:

```
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"]
```

2. Después de crear la CR de la aplicación para que coincida con tu entorno, aplica la CR. Por ejemplo:

```
kubectl apply -f maria-app.yaml
```

Pasos

1. Crea y aplica la definición de la aplicación usando uno de los siguientes ejemplos, reemplazando los valores entre corchetes por información de tu entorno. Puedes incluir espacios de nombres y recursos en la definición de la aplicación usando listas separadas por comas con los argumentos que se muestran en los ejemplos.

   Puedes usar opcionalmente una anotación cuando creas una app para especificar si la aplicación puede escribir en el sistema de archivos durante una snapshot. Esto solo es aplicable a aplicaciones definidas desde máquinas virtuales, como en entornos KubeVirt, donde los congelamientos del sistema de archivos ocurren antes de las snapshots. Si estableces la anotación en `true`, la aplicación ignora la configuración global y puede escribir en el sistema de archivos durante una snapshot. Si la estableces en `false`, la aplicación ignora la configuración global y el sistema de archivos se congela durante una snapshot. Si usas la anotación pero la aplicación no tiene máquinas

virtuales en la definición de la app, la anotación se ignora. Si no usas la anotación, la aplicación sigue la "ajuste global de congelación de Trident Protect".

Para especificar la anotación cuando uses la CLI para crear una aplicación, puedes usar el indicador `--annotation`.

- Crea la aplicación y usa la configuración global para el comportamiento de congelación del sistema de archivos:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace>
```

- Crea la aplicación y configura el ajuste local de la aplicación para el comportamiento de congelación del sistema de archivos:

```
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">
```

Puedes usar `--resource-filter-include` y `--resource-filter-exclude` flags para incluir o excluir recursos según `resourceSelectionCriteria` como grupo, tipo, versión, etiquetas, nombres y namespaces, como se muestra en el siguiente ejemplo:

```
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"]}]'
```

Protege las aplicaciones usando Trident Protect

Puedes proteger todas las apps gestionadas por Trident Protect tomando instantáneas y backups usando una política de protección automatizada o de forma ad hoc.

Puedes configurar Trident Protect para congelar y descongelar sistemas de archivos durante las operaciones de protección de datos. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".

Crea una instantánea a pedido

Puedes crear una instantánea a pedido en cualquier momento.

> Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Crear una instantánea usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: el nombre de Kubernetes de la aplicación para hacer snapshot.
   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se debe almacenar el contenido de la instantánea (metadatos).
   - **spec.reclaimPolicy**: (*opcional*) Define lo que pasa con la AppArchive de una instantánea cuando se elimina el CR de la instantánea. Esto significa que incluso cuando se establece en `Retain`, la instantánea se eliminará. Opciones válidas:
     - `Retain` (predeterminado)
     - `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. Después de rellenar el archivo `trident-protect-snapshot-cr.yaml` con los valores correctos, aplica la CR:

   ```
   kubectl apply -f trident-protect-snapshot-cr.yaml
   ```

Crea una instantánea usando la CLI

Pasos

1. Crea la instantánea, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

   ```
   tridentctl-protect create snapshot <my_snapshot_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> -n
   <application_namespace>
   ```

Crear un backup a pedido

Puedes hacer un backup de una app en cualquier momento.

> Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de backup s3 de larga duración. Si el token caduca durante la operación de backup, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Crea un backup usando una CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) el nombre de Kubernetes de la aplicación que quieres respaldar.
   - **spec.appVaultRef**: (*Obligatorio*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **spec.dataMover**: (*Opcional*) Una cadena que indica qué herramienta de backup se usará para la operación de backup. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **spec.reclaimPolicy**: (*Opcional*) Define qué sucede con una copia de seguridad al liberarla de su reclamación. Valores posibles:
     - `Delete`
     - `Retain` (predeterminado)
   - **spec.snapshotRef**: (*Opcional*): Nombre de la instantánea que se usará como origen del backup. Si no se proporciona, se creará una instantánea temporal y se hará un backup.

   Ejemplo de 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. Después de rellenar el archivo `trident-protect-backup-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Crea un backup usando la CLI

Pasos

1. Crea el backup, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

```
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>
```

Opcionalmente, puedes usar la `--full-backup` flag para especificar si un backup debe ser no incremental. Por defecto, todos los backups son incrementales. Cuando usas esta flag, el backup se vuelve no incremental. Lo mejor es hacer un backup completo de vez en cuando y luego backups incrementales entre los backups completos para minimizar el riesgo asociado con las restauraciones.

Anotaciones de backup compatibles

La siguiente tabla describe las anotaciones que puedes usar al crear una CR de backup:

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/backup-completo	cadena	Especifica si una copia de seguridad debe ser no incremental. Establece en `true` para crear una copia de seguridad no incremental. Es buena práctica hacer un backup completo periódicamente y luego hacer backups incrementales entre los backups completos para minimizar el riesgo asociado con las restauraciones.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	cadena	El tiempo máximo permitido para que se complete toda la operación de instantánea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	El tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	El tiempo máximo permitido para que se creen instantáneas de volumen.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle.	"1200" (20 minutos)

Crea un cronograma de protección de datos

Una política de protección protege una app creando snapshots, backups o ambos en una programación definida. Puedes elegir crear snapshots y backups cada hora, día, semana y mes, y puedes especificar la cantidad de copias que quieres conservar. Puedes programar un backup completo no incremental usando la anotación full-backup-rule. Por defecto, todos los backups son incrementales. Realizar un backup completo periódicamente, junto con backups incrementales entre medio, ayuda a reducir el riesgo asociado con las

restauraciones.

- Puedes crear programaciones solo para instantáneas configurando `backupRetention` en cero y `snapshotRetention` en un valor mayor que cero. Configurar `snapshotRetention` en cero significa que cualquier backup programado seguirá creando instantáneas, pero estas son temporales y se eliminan inmediatamente después de que el backup se complete.
- Los recursos con ámbito de clúster se incluyen en un backup, snapshot o clon si se hace referencia a ellos explícitamente en la definición de la aplicación o si tienen referencias a cualquiera de los espacios de nombres de la aplicación.

Crea un cronograma usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-schedule-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.dataMover**: (*Opcional*) Una cadena que indica qué herramienta de backup se usará para la operación de backup. Valores posibles (distingue mayúsculas de minúsculas):
     - `Restic`
     - `Kopia` (predeterminado)
   - **spec.applicationRef**: el nombre de Kubernetes de la aplicación que se va a respaldar.
   - **spec.appVaultRef**: (*Obligatorio*) El nombre del AppVault donde se debe almacenar el contenido de la copia de seguridad.
   - **spec.backupRetention**: (*Obligatorio*) El número de backups que se conservarán. Cero indica que no se deben crear backups (solo instantáneas).
   - **spec.backupReclaimPolicy**: (*Opcional*) Determina qué sucede con un backup si el backup CR se elimina durante su periodo de retención. Después del periodo de retención, los backups siempre se eliminan. Valores posibles (distingue mayúsculas de minúsculas):
     - `Retain` (predeterminado)
     - `Delete`
   - **spec.snapshotRetention**: (*Obligatorio*) El número de instantáneas que se conservarán. Cero indica que no se deben crear instantáneas.
   - **spec.snapshotReclaimPolicy**: (*Opcional*) Determina qué sucede con una instantánea si el CR de la instantánea se elimina durante su periodo de retención. Después del periodo de retención, las instantáneas siempre se eliminan. Valores posibles (distingue mayúsculas de minúsculas):
     - `Retain`
     - `Delete` (predeterminado)
   - **spec.granularity**: la frecuencia con la que debe ejecutarse la programación. Valores posibles, junto con los campos asociados obligatorios:
     - `Hourly` (requiere que especifiques `spec.minute`)
     - `Daily` (requiere que especifiques `spec.minute` y `spec.hour`)
     - `Weekly` (requiere que especifiques `spec.minute, spec.hour`, y `spec.dayOfWeek`)
     - `Monthly` (requiere que especifiques `spec.minute, spec.hour`, y `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Opcional*) El día del mes (1 - 31) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad está establecida en `Monthly`. El valor debe proporcionarse como una cadena.
   - **spec.dayOfWeek**: (*Opcional*) El día de la semana (0 - 7) en que debe ejecutarse la programación. Los valores 0 o 7 indican domingo. Este campo es obligatorio si la granularidad

está establecida en `Weekly`. El valor debe proporcionarse como una cadena.

- **spec.hour**: (*Opcional*) La hora del día (0 - 23) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad se establece en `Daily`, `Weekly` o `Monthly`. El valor debe proporcionarse como una cadena.
- **spec.minute**: (*Opcional*) El minuto de la hora (0 - 59) en que debe ejecutarse la programación. Este campo es obligatorio si la granularidad se establece en `Hourly`, `Daily`, `Weekly` o `Monthly`. El valor debe proporcionarse como una cadena.

Ejemplo de YAML para programación de backup y snapshots:

```
---
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"
```

Ejemplo de YAML para la programación solo de instantáneas:

```
---
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"
```

3. Después de rellenar el archivo `trident-protect-schedule-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crea un horario usando la CLI

Pasos

1. Crea el programa de protección, sustituyendo los valores entre corchetes por información de tu entorno. Por ejemplo:

> Puedes usar `tridentctl-protect create schedule --help` para ver información de ayuda detallada para este comando.

```
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>
```

Las siguientes banderas te dan un control adicional sobre tu horario:

- **Programación de backup completo**: Usa la bandera `--full-backup-rule` para programar backups completos no incrementales. Esta bandera solo funciona con `--granularity Daily`. Valores posibles:
  - `Always`: crea un backup completo cada día.
  - Días específicos de la semana: especifica uno o varios días separados por comas (por ejemplo, `"Monday,Thursday"`). Valores válidos: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

> La `--full-backup-rule` flag no funciona con granularidad horaria, semanal o mensual.

- **Planificaciones de solo instantáneas**: establece `--backup-retention 0` y especifica un valor mayor que cero para `--snapshot-retention`.

Anotaciones de horario compatibles

En la tabla siguiente se describen las anotaciones que puedes usar al crear un CR de planificación:

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/full-backup-rule	cadena	Especifica la regla para programar backups completos. Puedes configurarla en `Always` para un backup completo constante o personalizarla según tus necesidades. Por ejemplo, si eliges granularidad diaria, puedes especificar los días de la semana en los que debe hacerse el backup completo (por ejemplo, `"Monday,Thursday"`). Los valores válidos de días de la semana son: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Ten en cuenta que esta anotación solo se puede usar con programaciones que tengan `granularity` configurado en `Daily`.	No configurado (todos los backups son incrementales)
protect.trident.netapp.io/snapshot-completion-timeout	cadena	El tiempo máximo permitido para que se complete toda la operación de instantánea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadena	El tiempo máximo permitido para que las instantáneas de volumen alcancen el estado listo para usar.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadena	El tiempo máximo permitido para que se creen instantáneas de volumen.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle.	"1200" (20 minutos)

Borrar una instantánea

Elimina las instantáneas programadas o bajo demanda que ya no necesitas.

Pasos

1. Elimina el snapshot CR asociado con el snapshot:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Eliminar un backup

Elimina los backups programados o bajo demanda que ya no necesites.

Asegúrate de que la política de recuperación esté configurada en `Delete` para eliminar todos los datos de backup del almacenamiento de objetos. La configuración predeterminada de la política es `Retain` para evitar la pérdida accidental de datos. Si la política no se cambia a `Delete`, los datos de backup permanecerán en el almacenamiento de objetos y tendrás que eliminarlos manualmente.

Pasos

1. Elimina el backup CR asociado al backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Comprobar el estado de una operación de backup

Puedes usar la línea de comandos para comprobar el estado de un proceso de backup que está en curso, ha terminado o ha fallado.

Pasos

1. Usa el siguiente comando para recuperar el estado de la operación de backup, sustituyendo los valores entre corchetes por la información de tu entorno:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Habilita el backup y la restauración para las operaciones de azure-netapp-files (ANF)

Si has instalado Trident Protect, puedes activar la funcionalidad de backup y restauración con gestión eficiente del espacio para backends de almacenamiento que usan la clase de almacenamiento azure-netapp-files y que fueron creados antes de Trident 24.06. Esta funcionalidad funciona con volúmenes NFSv4 y no consume espacio adicional del capacity pool.

Antes de empezar

Asegúrate de lo siguiente:

- Has instalado Trident Protect.
- Has definido una aplicación en Trident Protect. Esta aplicación tendrá una funcionalidad de protección limitada hasta que completes este procedimiento.
- Has `azure-netapp-files` seleccionado como la clase de almacenamiento por defecto para tu backend de almacenamiento.

Expandir para pasos de configuración

1. Haz lo siguiente en Trident si el volumen ANF se creó antes de actualizar a Trident 24.10:

   a. Habilita el directorio de instantáneas para cada PV que esté basado en azure-netapp-files y asociado con la aplicación:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confirma que se ha habilitado el directorio de instantáneas para cada PV asociado:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Respuesta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Cuando el directorio de instantáneas no está habilitado, Trident Protect elige la funcionalidad de backup normal, que consume temporalmente espacio en el pool de capacidad durante el proceso de backup. En este caso, asegúrate de que hay suficiente espacio disponible en el pool de capacidad para crear un volumen temporal del tamaño del volumen que se está respaldando.

Resultado

La aplicación está lista para backup y restauración usando Trident Protect. Cada PVC también está disponible para que otras aplicaciones lo usen en backups y restauraciones.

Restaurar aplicaciones

Restaura aplicaciones usando Trident Protect

Puedes usar Trident Protect para restaurar tu aplicación desde una instantánea o backup. Restaurar desde una instantánea existente será más rápido cuando restaures la aplicación en el mismo clúster.

- Cuando restauras una aplicación, todos los ganchos de ejecución configurados para la aplicación se restauran junto con la app. Si hay un gancho de ejecución posterior a la restauración, se ejecuta automáticamente como parte de la operación de restauración.
- La restauración desde una copia de seguridad a un espacio de nombres diferente o al espacio de nombres original es compatible para los volúmenes qtree. Sin embargo, restaurar desde una instantánea a un espacio de nombres diferente o al espacio de nombres original no es compatible para los volúmenes qtree.
- Puedes usar la configuración avanzada para personalizar las operaciones de restauración. Para saber más, consulta "Usa la configuración avanzada de restauración de Trident Protect".

Restaura desde una copia de seguridad a un espacio de nombres diferente

Cuando restauras una copia de seguridad en un espacio de nombres diferente usando un CR de BackupRestore, Trident Protect restaura la aplicación en un nuevo espacio de nombres y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, crea copias de seguridad o instantáneas bajo demanda, o establece un programa de protección.

- Restaurar una copia de seguridad en un espacio de nombres diferente con recursos existentes no alterará ningún recurso que comparta nombres con los de la copia de seguridad. Para restaurar todos los recursos de la copia de seguridad, elimina y vuelve a crear el espacio de nombres de destino o restaura la copia de seguridad en un nuevo espacio de nombres.
- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea automáticamente espacios de nombres solo cuando usas la CLI.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
   - **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

     ```
     ---
     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
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Opcional*) Si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, añade un filtrado que incluya o excluya recursos marcados con etiquetas concretas:

   > Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y tiene un pod asociado, Trident Protect también restaurará el pod asociado.

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro

de cada elemento (group, kind, version) coinciden como una operación AND.

- **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
- **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
- **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
- **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

  Por ejemplo:

```
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. Después de rellenar el archivo `trident-protect-backup-restore-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Usa la CLI

Pasos

1. Restaura la copia de seguridad en un espacio de nombres diferente, reemplazando los valores entre corchetes por información de tu entorno. El argumento `namespace-mapping` usa espacios de nombres separados por dos puntos para mapear los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`. Por ejemplo:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restaura desde una copia de seguridad al espacio de nombres original

Puedes restaurar una copia de seguridad en el espacio de nombres original en cualquier momento.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

> Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-ipr-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.

     Por ejemplo:

     ```
     ---
     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. (*Opcional*) Si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, añade un filtrado que incluya o excluya recursos marcados con etiquetas concretas:

   > Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y tiene un pod asociado, Trident Protect también restaurará el pod asociado.

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro de cada elemento (group, kind, version) coinciden como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.

- **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
- **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

  Por ejemplo:

```
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. Después de rellenar el archivo `trident-protect-backup-ipr-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Usa la CLI

Pasos

1. Restaura la copia de seguridad en el espacio de nombres original, sustituyendo los valores entre corchetes por información de tu entorno. El argumento `backup` utiliza un espacio de nombres y un nombre de copia de seguridad en el formato `<namespace>/<name>`. Por ejemplo:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Restaurar desde una copia de seguridad en un clúster diferente

Puedes restaurar una copia de seguridad en un clúster diferente si hay algún problema con el clúster original.

- Cuando restauras copias de seguridad usando Kopia como el transportador de datos, puedes especificar opcionalmente anotaciones en el CR o usando la CLI para controlar el comportamiento del almacenamiento temporal que usa Kopia. Consulta la "Documentación de Kopia" para más información sobre las opciones que puedes configurar. Usa el comando `tridentctl-protect create --help` para más información sobre cómo especificar anotaciones con la Trident Protect CLI.
- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea automáticamente espacios de nombres solo cuando usas la CLI.

Antes de empezar

Asegúrate de que se cumplen los siguientes requisitos previos:

- El clúster de destino tiene Trident Protect instalado.
- El clúster de destino tiene acceso a la ruta del bucket de la misma AppVault que el clúster de origen, donde se almacena la copia de seguridad.
- Asegúrate de que tu entorno local puede conectarse al bucket de almacenamiento de objetos definido en el AppVault CR cuando ejecutes el comando `tridentctl-protect get appvaultcontent`. Si las restricciones de red impiden el acceso, ejecuta la CLI de Trident Protect desde un pod en el clúster de destino en su lugar.
- Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.
  - Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
  - Consulta "Documentación de AWS" para más información sobre las credenciales con los recursos de AWS.

Pasos

1. Comprueba la disponibilidad de AppVault CR en el clúster de destino usando el complemento CLI de Trident Protect:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

   Asegúrate de que el espacio de nombres previsto para la restauración de la aplicación existe en el clúster de destino.

2. Ver el contenido de la copia de seguridad de la AppVault disponible desde el clúster de destino:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Al ejecutar este comando se muestran las copias de seguridad disponibles en el AppVault, incluidos sus clústeres de origen, los nombres de las aplicaciones correspondientes, las marcas de tiempo y las rutas de archivo.

Ejemplo de salida:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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. Restaura la aplicación en el clúster de destino usando el nombre AppVault y la ruta de archivo:

Usa un CR

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-backup-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     Si BackupRestore CR no está disponible, puedes usar el comando mencionado en el paso 2 para ver el contenido de la copia de seguridad.

   - **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

     Por ejemplo:

```
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
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Después de rellenar el archivo `trident-protect-backup-restore-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Usa la CLI

1. Usa el siguiente comando para restaurar la aplicación, reemplazando los valores entre corchetes por la información de tu entorno. El argumento namespace-mapping utiliza espacios de nombres

separados por dos puntos para asignar los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato source1:dest1,source2:dest2. Por ejemplo:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restaurar desde una snapshot a un namespace diferente

Puedes restaurar datos de una instantánea usando un archivo de recurso personalizado (CR) ya sea en un espacio de nombres diferente o en el espacio de nombres de origen. Cuando restauras una instantánea en un espacio de nombres diferente usando un CR de SnapshotRestore, Trident Protect restaura la aplicación en un nuevo espacio de nombres y crea un CR de aplicación para la aplicación restaurada. Para proteger la aplicación restaurada, crea copias de seguridad o instantáneas bajo demanda, o establece un programa de protección.

- SnapshotRestore admite el atributo `spec.storageClassMapping`, pero solo cuando las clases de almacenamiento de origen y destino usan el mismo backend de almacenamiento. Si intentas restaurar en un `StorageClass` que usa un backend de almacenamiento diferente, la operación de restauración fallará.
- Cuando usas un CR para restaurar a un nuevo espacio de nombres, debes crear manualmente el espacio de nombres de destino antes de aplicar el CR. Trident Protect crea automáticamente espacios de nombres solo cuando usas la CLI.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

     ```
     ---
     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. (*Opcional*) Si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, añade un filtrado que incluya o excluya recursos marcados con etiquetas concretas:

   > Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y tiene un pod asociado, Trident Protect también restaurará el pod asociado.

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro

de cada elemento (group, kind, version) coinciden como una operación AND.

- **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
- **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
- **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.
- **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

  Por ejemplo:

```
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. Después de rellenar el archivo `trident-protect-snapshot-restore-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Usa la CLI

Pasos

1. Restaura la instantánea en un espacio de nombres diferente, reemplazando los valores entre corchetes con información de tu entorno.

   - El `snapshot`argumento usa un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>.
   - El `namespace-mapping` argumento utiliza espacios de nombres separados por dos puntos para

asignar los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`.

Por ejemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restaurar desde una snapshot al namespace original

Puedes restaurar una instantánea al espacio de nombres original en cualquier momento.

Antes de empezar

Asegúrate de que la caducidad del token de sesión de AWS sea suficiente para cualquier operación de restauración s3 de larga duración. Si el token caduca durante la operación de restauración, la operación puede fallar.

- Consulta "Documentación de AWS API" para más información sobre cómo comprobar la expiración del token de sesión actual.
- Consulta "Documentación de AWS IAM" para más información sobre las credenciales con los recursos de AWS.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-ipr-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     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. (*Opcional*) Si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, añade un filtrado que incluya o excluya recursos marcados con etiquetas concretas:

   > Trident Protect selecciona algunos recursos automáticamente debido a su relación con los recursos que tú seleccionas. Por ejemplo, si seleccionas un recurso de reclamación de volumen persistente y tiene un pod asociado, Trident Protect también restaurará el pod asociado.

   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `Include` o `Exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros resourceMatchers para definir los recursos que se van a incluir o excluir:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro de cada elemento (group, kind, version) coinciden como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
       - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.

- **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

  Por ejemplo:

```
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. Después de rellenar el archivo `trident-protect-snapshot-ipr-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Usa la CLI

Pasos

1. Restaura la instantánea en el espacio de nombres original, reemplazando los valores entre corchetes por información de tu entorno. Por ejemplo:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verifica el estado de una operación de restauración

Puedes usar la línea de comandos para comprobar el estado de una operación de restauración que está en curso, ha finalizado o ha fallado.

Pasos

1. Usa el siguiente comando para obtener el estado de la operación de restauración, reemplazando los valores entre corchetes por la información de tu entorno:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Usa la configuración avanzada de restauración de Trident Protect

Puedes personalizar las operaciones de restauración usando configuraciones avanzadas como anotaciones, configuración del espacio de nombres y opciones de almacenamiento para cumplir con tus requisitos específicos.

Anotaciones y etiquetas de namespace durante las operaciones de restauración y conmutación por error

Durante las operaciones de restauración y conmutación por error, las etiquetas y anotaciones en el espacio de nombres de destino se hacen coincidir con las etiquetas y anotaciones en el espacio de nombres de origen. Las etiquetas o anotaciones del espacio de nombres de origen que no existen en el espacio de nombres de destino se añaden, y cualquier etiqueta o anotación que ya exista se sobrescribe para que coincida con el valor del espacio de nombres de origen. Las etiquetas o anotaciones que existen solo en el espacio de nombres de destino permanecen sin cambios.

> Si usas Red Hat OpenShift, es importante que notes el papel fundamental de las anotaciones de espacios de nombres en entornos OpenShift. Las anotaciones de espacios de nombres aseguran que los pods restaurados sigan los permisos y configuraciones de seguridad apropiados definidos por las restricciones de contexto de seguridad (SCC) de OpenShift y puedan acceder a los volúmenes sin problemas de permisos. Para más información, consulta el "OpenShift documentación de restricciones de contexto de seguridad".

Puedes evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino configurando la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de realizar la restauración o la conmutación por error. Por ejemplo:

```
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
```

Al realizar una operación de restauración o conmutación por error, las anotaciones y etiquetas de espacios de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` se excluyen de la operación de restauración o conmutación por error. Asegúrate de que estos ajustes estén configurados durante la instalación inicial de Helm. Para saber más, consulta "Configura los ajustes adicionales del helm chart de Trident Protect".

Si instalaste la aplicación de origen usando Helm con la `--create-namespace` flag, se da un tratamiento especial a la clave de etiqueta `name`. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y uno de destino, cada uno con anotaciones y etiquetas diferentes. Puedes ver el estado del espacio de nombres de destino antes y después de la operación, y cómo se combinan o sobrescriben las anotaciones y etiquetas en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

La siguiente tabla ilustra el estado de los espacios de nombres de origen y destino de ejemplo antes de la operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-1 (fuente)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true"	• entorno=producción • compliance=hipaa • name=ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Después de la operación de restauración

La siguiente tabla ilustra el estado del espacio de nombres de destino de ejemplo después de la operación de restauración o conmutación por error. Se han añadido algunas claves, se han sobrescrito otras y la etiqueta `name` se ha actualizado para que coincida con el espacio de nombres de destino:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • entorno=producción • role=database

Campos compatibles

Esta sección describe los campos adicionales disponibles para las operaciones de restauración.

Asignación de clases de almacenamiento

El atributo `spec.storageClassMapping` define una asignación de una clase de almacenamiento presente en la aplicación de origen a una nueva clase de almacenamiento en el clúster de destino. Puedes usar esto cuando migres aplicaciones entre clústeres con diferentes clases de almacenamiento o cuando cambies el backend de almacenamiento para operaciones de BackupRestore.

Ejemplo:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Anotaciones compatibles

En esta sección se enumeran las anotaciones admitidas para configurar diversos comportamientos en el sistema. Si el usuario no establece explícitamente una anotación, el sistema usará el valor predeterminado.

Anotación	Tipo	Descripción	Valor predeterminado
protect.trident.netapp.io/data-mover-timeout-sec	cadena	El tiempo máximo (en segundos) permitido para que la operación de movimiento de datos se quede en pausa.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	cadena	El límite de tamaño máximo (en megabytes) para la caché de contenido de Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadena	Tiempo máximo (en segundos) de espera para que cualquier PersistentVolumeClaims (PVCs) recién creado alcance la `Bound` fase antes de que la operación falle. Aplica a todos los tipos de CR de restauración (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Usa un valor más alto si tu backend de almacenamiento o clúster suele requerir más tiempo.	"1200" (20 minutos)

Replica aplicaciones usando NetApp SnapMirror y Trident Protect

Con Trident Protect, puedes usar las capacidades de replicación asíncrona de NetApp SnapMirror para replicar datos y cambios de aplicaciones de un backend de almacenamiento a otro, en el mismo clúster o entre diferentes clústeres.

Anotaciones y etiquetas de namespace durante las operaciones de restauración y conmutación por error

Durante las operaciones de restauración y conmutación por error, las etiquetas y anotaciones en el espacio de nombres de destino se hacen coincidir con las etiquetas y anotaciones en el espacio de nombres de origen. Las etiquetas o anotaciones del espacio de nombres de origen que no existen en el espacio de nombres de destino se añaden, y cualquier etiqueta o anotación que ya exista se sobrescribe para que coincida con el valor del espacio de nombres de origen. Las etiquetas o anotaciones que existen solo en el espacio de nombres de destino permanecen sin cambios.

> Si usas Red Hat OpenShift, es importante que notes el papel fundamental de las anotaciones de espacios de nombres en entornos OpenShift. Las anotaciones de espacios de nombres aseguran que los pods restaurados sigan los permisos y configuraciones de seguridad apropiados definidos por las restricciones de contexto de seguridad (SCC) de OpenShift y puedan acceder a los volúmenes sin problemas de permisos. Para más información, consulta el "OpenShift documentación de restricciones de contexto de seguridad".

Puedes evitar que se sobrescriban anotaciones específicas en el espacio de nombres de destino configurando la variable de entorno de Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de realizar la restauración o la conmutación por error. Por ejemplo:

```
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
```

> Al realizar una operación de restauración o conmutación por error, las anotaciones y etiquetas de espacios de nombres especificadas en `restoreSkipNamespaceAnnotations` y `restoreSkipNamespaceLabels` se excluyen de la operación de restauración o conmutación por error. Asegúrate de que estos ajustes estén configurados durante la instalación inicial de Helm. Para saber más, consulta "Configura los ajustes adicionales del helm chart de Trident Protect".

Si instalaste la aplicación de origen usando Helm con la `--create-namespace` flag, se da un tratamiento especial a la clave de etiqueta `name`. Durante el proceso de restauración o conmutación por error, Trident Protect copia esta etiqueta al espacio de nombres de destino, pero actualiza el valor al valor del espacio de nombres de destino si el valor del origen coincide con el espacio de nombres de origen. Si este valor no coincide con el espacio de nombres de origen, se copia al espacio de nombres de destino sin cambios.

Ejemplo

El siguiente ejemplo presenta un espacio de nombres de origen y uno de destino, cada uno con anotaciones y etiquetas diferentes. Puedes ver el estado del espacio de nombres de destino antes y después de la operación, y cómo se combinan o sobrescriben las anotaciones y etiquetas en el espacio de nombres de destino.

Antes de la operación de restauración o conmutación por error

La siguiente tabla ilustra el estado de los espacios de nombres de origen y destino de ejemplo antes de la

operación de restauración o conmutación por error:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-1 (fuente)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true"	• entorno=producción • compliance=hipaa • name=ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Después de la operación de restauración

La siguiente tabla ilustra el estado del espacio de nombres de destino de ejemplo después de la operación de restauración o conmutación por error. Se han añadido algunas claves, se han sobrescrito otras y la etiqueta `name` se ha actualizado para que coincida con el espacio de nombres de destino:

Espacio de nombres	Anotaciones	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "valoractualizado" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • entorno=producción • role=database

Puedes configurar Trident Protect para congelar y descongelar sistemas de archivos durante las operaciones de protección de datos. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".

Ganchos de ejecución durante la conmutación por error y las operaciones inversas

Cuando uses la relación AppMirror para proteger tu aplicación, hay comportamientos específicos relacionados con los hooks de ejecución que deberías conocer durante las operaciones de conmutación por error y reversa.

- Durante la conmutación por error, los ganchos de ejecución se copian automáticamente del clúster de origen al clúster de destino. No necesitas volver a crearlos manualmente. Después de la conmutación por error, los ganchos de ejecución están presentes en la aplicación y se ejecutarán durante cualquier acción relevante.
- Durante la resincronización inversa o inversa, se eliminan todos los execution hooks existentes en la aplicación. Cuando la source application se convierte en la destination application, estos execution hooks no son válidos y se eliminan para evitar su ejecución.

Para obtener más información sobre los ganchos de ejecución, consulta "Administra los hooks de ejecución de Trident Protect".

Configura una relación de replicación

Configurar una relación de replicación implica lo siguiente:

- Elegir con qué frecuencia quieres que Trident Protect tome una instantánea de la app (que incluye los recursos de Kubernetes de la app, así como las instantáneas de volumen para cada uno de los volúmenes de la app)
- Elegir el calendario de replicación (incluye los recursos de Kubernetes y también los datos de volumen persistente)
- Configurar la hora para que se tome la instantánea

Pasos

1. En el clúster de origen, crea un AppVault para la aplicación de origen. Dependiendo de tu proveedor de almacenamiento, modifica un ejemplo en "AppVault recursos personalizados" para adaptarlo a tu entorno:

Crea un AppVault usando un CR

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-appvault-primary-source.yaml`).

b. Configura los siguientes atributos:

- **metadata.name**: (*Required*) El nombre del recurso personalizado AppVault. Toma nota del nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.providerConfig**: (*Requerido*) Almacena la configuración necesaria para acceder a AppVault usando el proveedor especificado. Elige un bucketName y cualquier otro detalle necesario para tu proveedor. Toma nota de los valores que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulta "AppVault recursos personalizados" para ver ejemplos de CR de AppVault con otros proveedores.
- **spec.providerCredentials**: (*Obligatorio*) Almacena referencias a cualquier credencial necesaria para acceder a AppVault utilizando el proveedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Required*) Indica que el valor de la credencial debe venir de un secreto.
    - **key**: (*Obligatorio*) La clave válida del secreto a seleccionar.
    - **nombre**: (*Required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatorio*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatorio*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Valores posibles:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Después de rellenar el archivo `trident-protect-appvault-primary-source.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n trident-protect
```

Crea un AppVault usando la CLI

a. Crea el AppVault, reemplazando los valores entre corchetes con información de tu entorno:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. En el clúster de origen, crea la CR de la aplicación de origen:

Crea la aplicación fuente usando una CR

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-app-source.yaml`).

b. Configura los siguientes atributos:

- **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de la aplicación. Toma nota del nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
- **spec.includedNamespaces**: (*Obligatorio*) Una matriz de espacios de nombres y etiquetas asociadas. Usa los nombres de los espacios de nombres y, si quieres, limita el alcance de los espacios de nombres con etiquetas para especificar los recursos que existen en los espacios de nombres que aparecen aquí. El espacio de nombres de la aplicación debe formar parte de esta matriz.

  Ejemplo 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. Después de rellenar el archivo `trident-protect-app-source.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Crea la aplicación de origen usando la CLI

a. Crea la aplicación fuente. Por ejemplo:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Opcionalmente, en el clúster de origen, toma una instantánea de la aplicación de origen. Esta instantánea se usa como base para la aplicación en el clúster de destino. Si te saltas este paso, tendrás que esperar a que se ejecute la siguiente instantánea programada para tener una instantánea reciente. Para crear una instantánea bajo demanda, consulta "Crea una instantánea a pedido".

4. En el clúster de origen, crea la programación de replicación CR:

Además del programa que se proporciona a continuación, se recomienda crear un programa de instantáneas diarias independiente con un período de retención de 7 días para mantener una instantánea común entre clústeres de ONTAP emparejados. Esto garantiza que las instantáneas estén disponibles hasta por 7 días, pero el período de retención se puede personalizar según las necesidades del usuario.

Si se produce una conmutación por error, el sistema puede usar estas instantáneas hasta por 7 días para operaciones inversas. Este enfoque hace que el proceso inverso sea más rápido y eficiente porque solo se transferirán los cambios hechos desde la última instantánea, no todos los datos.

Si un programa existente para la aplicación ya cumple con los requisitos de retención deseados, no se requieren programaciones adicionales.

Crea el programa de replicación usando una CR

a. Crea un programa de replicación para la aplicación de origen:

   i. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-schedule.yaml`).

   ii. Configura los siguientes atributos:

   - **metadata.name**: (*Obligatorio*) El nombre del recurso personalizado de schedule.
   - **spec.appVaultRef**: (*Obligatorio*) Este valor debe coincidir con el campo metadata.name de la AppVault para la aplicación de origen.
   - **spec.applicationRef**: (*Obligatorio*) Este valor debe coincidir con el campo metadata.name del CR de la aplicación de origen.
   - **spec.backupRetention**: (*Obligatorio*) Este campo es obligatorio y el valor debe establecerse en 0.
   - **spec.enabled**: debe establecerse en true.
   - **spec.granularity**: debe establecerse en `Custom`.
   - **spec.recurrenceRule**: define una fecha de inicio en tiempo UTC y un intervalo de recurrencia.
   - **spec.snapshotRetention**: debe establecerse en 2.

     Ejemplo de 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. Después de rellenar el archivo `trident-protect-schedule.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

Crea la programación de replicación usando la CLI

a. Crea el programa de replicación, reemplazando los valores entre corchetes con información de tu entorno:

```
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>
```

Ejemplo:

```
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. En el clúster de destino, crea una CR de aplicación de origen AppVault que sea idéntica a la CR AppVault que aplicaste en el clúster de origen y asígnale un nombre (por ejemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplica la CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Crea un CR de AppVault de destino para la aplicación de destino en el clúster de destino. Dependiendo de tu proveedor de almacenamiento, modifica un ejemplo en "AppVault recursos personalizados" para adaptarlo a tu entorno:
   a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configura los siguientes atributos:
      - **metadata.name**: (*Required*) El nombre del recurso personalizado AppVault. Toma nota del nombre que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a este valor.
      - **spec.providerConfig**: (*Obligatorio*) Almacena la configuración necesaria para acceder a AppVault usando el proveedor especificado. Elige un `bucketName` y cualquier otro detalle necesario para tu proveedor. Toma nota de los valores que elijas, porque otros archivos CR necesarios para una relación de replicación hacen referencia a estos valores. Consulta "AppVault recursos

personalizados" para ver ejemplos de CR de AppVault con otros proveedores.

- **spec.providerCredentials**: (*Obligatorio*) Almacena referencias a cualquier credencial necesaria para acceder a AppVault utilizando el proveedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Required*) Indica que el valor de la credencial debe venir de un secreto.
    - **key**: (*Obligatorio*) La clave válida del secreto a seleccionar.
    - **nombre**: (*Required*) Nombre del secreto que contiene el valor para este campo. Debe estar en el mismo espacio de nombres.
  - **spec.providerCredentials.secretAccessKey**: (*Obligatorio*) La clave de acceso utilizada para acceder al proveedor. El **name** debe coincidir con **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obligatorio*) determina qué proporciona la copia de seguridad; por ejemplo, NetApp ONTAP S3, S3 genérico, Google Cloud o Microsoft Azure. Valores posibles:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Después de rellenar el archivo `trident-protect-appvault-secondary-destination.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. En el clúster de destino, crea un archivo CR de AppMirrorRelationship.

> Al usar una CR, crea manualmente el espacio de nombres de destino antes de aplicarla. Trident Protect crea espacios de nombres automáticamente solo cuando usas la CLI.

Crea un AppMirrorRelationship usando un CR

a. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-relationship.yaml`).

b. Configura los siguientes atributos:

- **metadata.name:** (Obligatorio) El nombre del recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Obligatorio*) Este valor debe coincidir con el nombre de AppVault para la aplicación de destino en el clúster de destino.
- **spec.namespaceMapping**: (*Obligatorio*) Los espacios de nombres de destino y de origen deben coincidir con el espacio de nombres de la aplicación definido en el CR de la aplicación respectiva.
- **spec.sourceAppVaultRef**: (*Obligatorio*) Este valor debe coincidir con el nombre de AppVault para la aplicación de origen.
- **spec.sourceApplicationName**: (*Obligatorio*) Este valor debe coincidir con el nombre de la aplicación de origen que definiste en el CR de la aplicación de origen.
- **spec.sourceApplicationUID**: (Obligatorio) Este valor debe coincidir con el UID de la aplicación de origen que definiste en el CR de la aplicación de origen.
- **spec.storageClassName**: (*opcional*) elige el nombre de una clase de almacenamiento válida en el clúster. La clase de almacenamiento debe estar vinculada a una máquina virtual de almacenamiento ONTAP que esté emparejada con el entorno de origen. Si no se proporciona la clase de almacenamiento, se usará la clase de almacenamiento predeterminada del clúster.
- **spec.recurrenceRule**: define una fecha de inicio en tiempo UTC y un intervalo de recurrencia.

  Ejemplo de 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. Después de rellenar el archivo `trident-protect-relationship.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Crea un AppMirrorRelationship usando la CLI

a. Crea y aplica el objeto AppMirrorRelationship, reemplazando los valores entre corchetes con información de tu entorno:

```
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>
```

Ejemplo:

```
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. (*Opcional*) En el clúster de destino, revisa el estado y el estatus de la relación de replicación:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Conmutar al clúster de destino

Con Trident Protect, puedes conmutar por error las aplicaciones replicadas a un clúster de destino. Este procedimiento detiene la relación de replicación y pone la app en línea en el clúster de destino. Trident Protect no detiene la app en el clúster de origen si estaba operativa.

Pasos

1. En el clúster de destino, edita el archivo CR AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambia el valor de **spec.desiredState** a `Promoted`.
2. Guarda el archivo CR.
3. Aplica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Opcional*) Crea cualquier programación de protección que necesites en la aplicación conmutada por error.
5. (*Opcional*) Verifica el estado y el estatus de la relación de replicación:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Resincronizar una relación de replicación conmutada por error

La operación de resincronización restablece la relación de replicación. Después de realizar una operación de resincronización, la aplicación de origen original se convierte en la aplicación en ejecución y cualquier cambio hecho en la aplicación en ejecución en el clúster de destino se descarta.

El proceso detiene la app en el clúster de destino antes de restablecer la replicación.

Cualquier dato escrito en la aplicación de destino durante la conmutación por error se perderá.

Pasos

1. Opcional: en el clúster de origen, crea una instantánea de la aplicación de origen. Esto asegura que se capturen los últimos cambios del clúster de origen.
2. En el clúster de destino, edita el archivo CR de AppMirrorRelationship (por ejemplo, `trident-protect-relationship.yaml`) y cambia el valor de spec.desiredState a `Established`.
3. Guarda el archivo CR.
4. Aplica la CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Si creaste alguna programación de protección en el clúster de destino para proteger la aplicación que se transfirió tras el fallo, elimínala. Cualquier programación que quede provoca fallos en las instantáneas del volumen.

Resincroniza de forma inversa una relación de replicación que ha fallado

Cuando haces una resincronización inversa de una relación de replicación fallida, la aplicación de destino se convierte en la aplicación de origen y la de origen en la de destino. Los cambios hechos en la aplicación de destino durante el failover se mantienen.

Pasos

1. En el clúster de destino original, elimina el CR AppMirrorRelationship. Esto hace que el destino se convierta en el origen. Si quedan programas de protección en el nuevo clúster de destino, elimínalos.
2. Configura una relación de replicación aplicando los archivos CR que usaste originalmente para configurar la relación en los clústeres opuestos.
3. Asegúrate de que el nuevo destino (clúster de origen original) esté configurado con ambos CR de AppVault.
4. Configura una relación de replicación en el clúster opuesto, configurando los valores para la dirección inversa.

Invertir la dirección de replicación de la aplicación

Cuando inviertes la dirección de replicación, Trident Protect mueve la aplicación al backend de almacenamiento de destino mientras sigue replicando de vuelta al backend de almacenamiento de origen original. Trident Protect detiene la aplicación de origen y replica los datos al destino antes de hacer el failover a la app de destino.

En esta situación, estás intercambiando el clúster de origen y el clúster de destino.

Pasos

1. En el clúster de origen, crea una instantánea de apagado:

Crea una instantánea de apagado usando un CR

a. Desactiva las programaciones de la política de protección para la aplicación de origen.

b. Crea un archivo ShutdownSnapshot CR:

   i. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configura los siguientes atributos:

   - **metadata.name**: (*Requerido*) El nombre del recurso personalizado.
   - **spec.AppVaultRef**: (*Required*) Este valor debe coincidir con el campo metadata.name del AppVault para la aplicación de origen.
   - **spec.ApplicationRef**: (*Required*) Este valor debe coincidir con el campo metadata.name del archivo CR de la aplicación de origen.

     Ejemplo de 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. Después de rellenar el archivo `trident-protect-shutdownsnapshot.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Crea una instantánea de apagado usando la CLI

a. Crea la instantánea de apagado, reemplazando los valores entre corchetes por información de tu entorno. Por ejemplo:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. En el clúster de origen, después de que se complete la instantánea de apagado, obtén el estado de la instantánea de apagado:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. En el clúster de origen, encuentra el valor de **shutdownsnapshot.status.appArchivePath** usando el siguiente comando y apunta la última parte de la ruta del archivo (también llamada basename; esto será todo lo que esté después de la última barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Realiza una conmutación por error del nuevo clúster de destino al nuevo clúster de origen, con el siguiente cambio:

> ⓘ En el paso 2 del procedimiento de conmutación por error, incluye el campo `spec.promotedSnapshot` en el archivo AppMirrorRelationship CR y pon su valor en el nombre base que anotaste en el paso 3 de arriba.

5. Realiza los pasos de resincronización inversa en Resincroniza de forma inversa una relación de replicación que ha fallado.
6. Habilita los calendarios de protección en el nuevo clúster de origen.

Resultado

Las siguientes acciones ocurren debido a la replicación inversa:

- Se toma una instantánea de los recursos de Kubernetes de la aplicación fuente original.
- Los pods de la app fuente original se detienen de forma gradual eliminando los recursos de Kubernetes de la app (dejando los PVCs y PVs en su lugar).
- Después de que los pods se apagan, se toman instantáneas de los volúmenes de la app y se replican.
- Las relaciones de SnapMirror se rompen, lo que deja los volúmenes de destino listos para lectura/escritura.
- Los recursos de Kubernetes de la app se restauran desde la instantánea previa al apagado, usando los datos de volumen replicados después de que la app de origen original se apagara.
- La replicación se restablece en sentido inverso.

Devuelve las aplicaciones al clúster de origen

Con Trident Protect, puedes lograr la "recuperación" después de una operación de conmutación por error usando la siguiente secuencia de operaciones. En este flujo de trabajo para restaurar la dirección de replicación original, Trident Protect replica (resincroniza) cualquier cambio de la aplicación de vuelta a la aplicación de origen original antes de invertir la dirección de replicación.

Este proceso parte de una relación que ha completado una conmutación por error a un destino e implica los siguientes pasos:

- Empieza con un estado conmutado por error.
- Resincroniza de forma inversa la relación de replicación.

> No realices una operación de resincronización normal, ya que esto descartará los datos escritos en el clúster de destino durante el procedimiento de conmutación por error.

- Invierte la dirección de replicación.

Pasos

1. Realiza los pasos de Resincroniza de forma inversa una relación de replicación que ha fallado.
2. Realiza los pasos de Invertir la dirección de replicación de la aplicación.

Eliminar una relación de replicación

Puedes eliminar una relación de replicación en cualquier momento. Cuando eliminas la relación de replicación de aplicaciones, el resultado son dos aplicaciones separadas sin relación entre ellas.

Pasos

1. En el clúster de destino actual, elimina el CR AppMirrorRelationship:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migra aplicaciones usando Trident Protect

Puedes migrar tus aplicaciones entre clústeres o a diferentes clases de almacenamiento restaurando los datos de backup.

> Cuando migras una aplicación, todos los ganchos de ejecución configurados para la aplicación se migran con la app. Si hay un gancho de ejecución posterior a la restauración, se ejecuta automáticamente como parte de la operación de restauración.

Operaciones de backup y restauración

Para realizar operaciones de backup y restauración en los siguientes escenarios, puedes automatizar tareas específicas de backup y restauración.

Clonar en el mismo clúster

Para clonar una aplicación en el mismo clúster, crea una instantánea o backup y restaura los datos en el mismo clúster.

Pasos

1. Debe realizar una de las siguientes acciones:
   a. "Crear una instantánea".
   b. "Crear un backup".
2. En el mismo clúster, haz una de las siguientes cosas, dependiendo de si creaste una instantánea o un backup:

a. "Restaura tus datos desde la instantánea".

b. "Restaura tus datos desde el backup".

Clonar a clúster diferente

Para clonar una aplicación en un clúster diferente (realizar un clon entre clústeres), crea un backup en el clúster de origen y luego restaura el backup en un clúster diferente. Asegúrate de que Trident Protect está instalado en el clúster de destino.

> Puedes replicar una aplicación entre diferentes clústeres usando "Replicación de SnapMirror".

Pasos

1. "Crear un backup".
2. Asegúrate de que el AppVault CR para el bucket de almacenamiento de objetos que contiene el backup se ha configurado en el clúster de destino.
3. En el clúster de destino, "restaura tus datos desde el backup".

Migra aplicaciones de una clase de almacenamiento a otra clase de almacenamiento

Puedes migrar aplicaciones de una clase de almacenamiento a otra diferente restaurando un backup en la clase de almacenamiento de destino.

Por ejemplo (excluyendo los secretos del restore 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
```

Restaura la instantánea usando una CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-snapshot-restore-cr.yaml`.
2. En el archivo que creaste, configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la instantánea. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Required*) El nombre de AppVault donde se almacenan los contenidos de la instantánea.
   - **spec.namespaceMapping**: la asignación del espacio de nombres de origen de la operación de restauración al espacio de nombres de destino. Reemplaza `my-source-namespace` y `my-destination-namespace` con información de tu entorno.

     ```
     ---
     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. Opcionalmente, si necesitas seleccionar solo ciertos recursos de la aplicación para restaurar, agrega un filtrado que incluya o excluya recursos marcados con etiquetas particulares:
   - **resourceFilter.resourceSelectionCriteria**: (Obligatorio para el filtrado) Usa `include or exclude` para incluir o excluir un recurso definido en resourceMatchers. Agrega los siguientes parámetros de resourceMatchers para definir los recursos que se incluirán o excluirán:
     - **resourceFilter.resourceMatchers**: una matriz de objetos resourceMatcher. Si defines múltiples elementos en esta matriz, coinciden como una operación OR y los campos dentro de cada elemento (group, kind, version) coinciden como una operación AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo del recurso a filtrar.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo del recurso a filtrar.
       - **resourceMatchers[].version**: (*Opcional*) Versión del recurso a filtrar.

- **resourceMatchers[].names**: (*Opcional*) Nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].namespaces**: (*Opcional*) Espacios de nombres en el campo metadata.name de Kubernetes del recurso que se va a filtrar.
- **resourceMatchers[].labelSelectors**: (*opcional*) Cadena de selector de etiqueta en el campo metadata.name de Kubernetes del recurso, como se define en "Documentación de Kubernetes". Por ejemplo: `"trident.netapp.io/os=linux"`.

  Por ejemplo:

```
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. Después de rellenar el archivo `trident-protect-snapshot-restore-cr.yaml` con los valores correctos, aplica la CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaura la instantánea usando la CLI

Pasos

1. Restaura la instantánea en un espacio de nombres diferente, reemplazando los valores entre corchetes con información de tu entorno.

   - El `snapshot`argumento usa un espacio de nombres y un nombre de instantánea en el formato `<namespace>/<name>.
   - El `namespace-mapping` argumento utiliza espacios de nombres separados por dos puntos para asignar los espacios de nombres de origen a los espacios de nombres de destino correctos en el formato `source1:dest1,source2:dest2`.

     Por ejemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Administra los hooks de ejecución de Trident Protect

Un gancho de ejecución es una acción personalizada que puedes configurar para que se ejecute junto con una operación de protección de datos de una app gestionada. Por ejemplo, si tienes una app de base de datos, puedes usar un gancho de ejecución para pausar todas las transacciones de la base de datos antes de una instantánea y reanudar las transacciones después de que la instantánea esté completa. Esto garantiza instantáneas consistentes con las aplicaciones.

Tipos de hooks de ejecución

Trident Protect admite los siguientes tipos de ganchos de ejecución, según cuándo pueden ejecutarse:

- Previa a la instantánea
- Posterior a la instantánea
- Previa a la copia de seguridad
- Posterior a la copia de seguridad
- Post-restauración
- Después de la conmutación por error

Orden de ejecución

Cuando se ejecuta una operación de protección de datos, los eventos de hook de ejecución ocurren en el siguiente orden:

1. Cualquier gancho personalizado aplicable de ejecución previa a la operación se ejecuta en los contenedores apropiados. Puedes crear y ejecutar tantos ganchos personalizados de ejecución previa a la operación como necesites, pero el orden de ejecución de estos ganchos antes de la operación no está garantizado ni es configurable.
2. Ocurre la congelación del sistema de archivos, si aplica. "Conoce más sobre cómo configurar la congelación del sistema de archivos con Trident Protect".
3. Se realiza la operación de protección de datos.
4. Los sistemas de archivos congelados se descongelan, si aplica.
5. Todos los hooks personalizados de ejecución post-operación aplicables se ejecutan en los contenedores apropiados. Puedes crear y ejecutar tantos hooks personalizados de ejecución post-operación como necesites, pero el orden de ejecución de estos hooks después de la operación no está garantizado ni es configurable.

Si creas varios ganchos de ejecución del mismo tipo (por ejemplo, pre-snapshot), el orden de ejecución de esos ganchos no está garantizado. Sin embargo, el orden de ejecución de ganchos de diferentes tipos sí está garantizado. Por ejemplo, este es el orden de ejecución de una configuración que tiene todos los diferentes

tipos de ganchos:

1. Ganchos previos a la instantánea ejecutados
2. Ganchos post-snapshot ejecutados
3. Ganchos previos a la copia de seguridad ejecutados
4. Ganchos posteriores a la copia de seguridad ejecutados

El ejemplo de orden anterior solo se aplica cuando ejecutas una copia de seguridad que no usa una instantánea existente.

Siempre debes probar tus scripts de ganchos de ejecución antes de habilitarlos en un entorno de producción. Puedes usar el comando 'kubectl exec' para probar cómodamente los scripts. Después de habilitar los ganchos de ejecución en un entorno de producción, prueba las instantáneas y copias de seguridad resultantes para asegurarte de que son consistentes. Puedes hacer esto clonando la app en un espacio de nombres temporal, restaurando la instantánea o copia de seguridad y luego probando la app.

Si un hook de ejecución previo a la instantánea añade, cambia o elimina recursos de Kubernetes, esos cambios se incluyen en la instantánea o backup y en cualquier operación de restauración posterior.

Notas importantes sobre los ganchos de ejecución personalizados

Ten en cuenta lo siguiente al planificar los hooks de ejecución para tus apps.

- Un gancho de ejecución debe usar un script para realizar acciones. Muchos ganchos de ejecución pueden hacer referencia al mismo script.
- Trident Protect requiere que los scripts que usan los hooks de ejecución estén escritos en formato de scripts de shell ejecutables.
- El tamaño del script está limitado a 96KB.
- Trident Protect utiliza la configuración de los execution hook y cualquier criterio de coincidencia para determinar qué hooks son aplicables a una operación de snapshot, backup o restore.

Debido a que los ganchos de ejecución a menudo reducen o deshabilitan por completo la funcionalidad de la aplicación contra la que se ejecutan, siempre debes intentar minimizar el tiempo que tardan en ejecutarse tus ganchos de ejecución personalizados. Si inicias una operación de copia de seguridad o instantánea con ganchos de ejecución asociados pero luego la cancelas, los ganchos aún pueden ejecutarse si la operación de copia de seguridad o instantánea ya ha comenzado. Esto significa que la lógica utilizada en un gancho de ejecución posterior a la copia de seguridad no puede asumir que la copia de seguridad se completó.

Filtros de ejecución hook

Cuando añades o editas un gancho de ejecución para una aplicación, puedes añadir filtros al gancho de ejecución para gestionar en qué contenedores coincidirá el gancho. Los filtros son útiles para aplicaciones que usan la misma imagen de contenedor en todos los contenedores, pero podrían usar cada imagen para un propósito diferente (como Elasticsearch). Los filtros te permiten crear escenarios donde los ganchos de ejecución se ejecutan en algunos, pero no necesariamente en todos los contenedores idénticos. Si creas varios filtros para un solo gancho de ejecución, se combinan con un operador lógico AND. Puedes tener hasta 10 filtros activos por gancho de ejecución.

Cada filtro que añades a un gancho de ejecución usa una expresión regular para hacer coincidir los contenedores en tu clúster. Cuando un gancho coincide con un contenedor, el gancho ejecutará su script asociado en ese contenedor. Las expresiones regulares para los filtros usan la sintaxis Regular Expression 2 (RE2), que no permite crear un filtro que excluya contenedores de la lista de coincidencias. Para información sobre la sintaxis que Trident Protect admite para expresiones regulares en los filtros de ganchos de ejecución, consulta "Compatibilidad con la sintaxis de Regular Expression 2 (RE2)".

> Si añades un filtro de espacio de nombres a un gancho de ejecución que se ejecuta después de una operación de restauración o clonación, y el origen y el destino de la restauración o clonación están en espacios de nombres diferentes, el filtro de espacio de nombres solo se aplica al espacio de nombres de destino.

Ejemplos de execution hook

Visita el "Proyecto NetApp Verda GitHub" para descargar ganchos de ejecución reales para apps populares como Apache Cassandra y Elasticsearch. También puedes ver ejemplos y obtener ideas para estructurar tus propios ganchos de ejecución personalizados.

Crear un gancho de ejecución

Puedes crear un gancho de ejecución personalizado para una app usando Trident Protect. Necesitas tener permisos de Owner, Admin o Member para crear ganchos de ejecución.

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-hook.yaml`.
2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) el nombre de Kubernetes de la aplicación para la que quieres ejecutar el gancho de ejecución.
   - **spec.stage**: (*Obligatorio*) una cadena que indica en qué etapa de la acción debe ejecutarse el gancho de ejecución. Valores posibles:
     - Pre
     - Publicar
   - **spec.action**: (*Required*) Una cadena que indica qué acción realizará el gancho de ejecución, suponiendo que coincida cualquier filtro de gancho de ejecución especificado. Valores posibles:
     - Snapshot
     - Copia de seguridad
     - Restaurar
     - Conmutación por error
   - **spec.enabled**: (*Opcional*) Indica si este execution hook está habilitado o deshabilitado. Si no se especifica, el valor predeterminado es true.
   - **spec.hookSource**: (*Required*) Una cadena que contiene el script de hook codificado en base64.
   - **spec.timeout**: (*Opcional*) Un número que define cuánto tiempo en minutos se permite que se ejecute el hook de ejecución. El valor mínimo es 1 minuto y el valor por defecto es 25 minutos si no se especifica.
   - **spec.arguments**: (*Opcional*) Una lista YAML de argumentos que puedes especificar para el gancho de ejecución.
   - **spec.matchingCriteria**: (*Opcional*) Una lista opcional de pares clave-valor de criterios, cada par constituye un filtro de gancho de ejecución. Puedes añadir hasta 10 filtros por gancho de ejecución.
   - **spec.matchingCriteria.type**: (*Opcional*) una cadena que identifica el tipo de filtro de gancho de ejecución. Valores posibles:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Opcional*) Una cadena o expresión regular que identifica el valor del filtro de ejecución.

     Ejemplo de 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. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Usa la CLI

Pasos

1. Crea el gancho de ejecución, sustituyendo los valores entre corchetes por información de tu entorno. Por ejemplo:

```
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>
```

Ejecuta manualmente un gancho de ejecución

Puedes ejecutar manualmente un gancho de ejecución con fines de prueba o si necesitas volver a ejecutar el gancho manualmente después de un fallo. Necesitas tener permisos de Owner, Admin o Member para ejecutar manualmente ganchos de ejecución.

Ejecutar manualmente un hook de ejecución consiste en dos pasos básicos:

1. Crea una copia de seguridad de recursos, que recoge los recursos y crea una copia de seguridad de ellos, determinando dónde se ejecutará el hook
2. Ejecuta el hook de ejecución contra la copia de seguridad

Paso 1: crea una copia de seguridad de los recursos

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-resource-backup.yaml`.
2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Required*) El nombre de Kubernetes de la aplicación para la que crear la copia de seguridad de recursos.
   - **spec.appVaultRef**: (*Required*) el nombre del AppVault donde se almacena el contenido de la copia de seguridad.
   - **spec.appArchivePath**: la ruta dentro de AppVault donde se almacena el contenido de la copia de seguridad. Puedes usar el siguiente comando para encontrar esta ruta:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     Ejemplo de 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. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Usa la CLI

Pasos

1. Crea el backup, reemplazando los valores entre corchetes con información de tu entorno. Por ejemplo:

```
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. Consulta el estado de la copia de seguridad. Puedes usar este comando de ejemplo repetidamente hasta que la operación termine:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verifica que la copia de seguridad se realizó correctamente:

```
kubectl describe resourcebackup <my_backup_name>
```

Paso 2: ejecuta el gancho de ejecución

Usa un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y asígnale el nombre `trident-protect-hook-run.yaml`.
2. Configura los siguientes atributos para que coincidan con tu entorno de Trident Protect y la configuración del clúster:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.applicationRef**: (*Obligatorio*) asegúrate de que este valor coincide con el nombre de la aplicación del CR ResourceBackup que creaste en el paso 1.
   - **spec.appVaultRef**: (*Obligatorio*) Asegúrate de que este valor coincide con el appVaultRef del ResourceBackup CR que creaste en el paso 1.
   - **spec.appArchivePath**: asegúrate de que este valor coincide con el appArchivePath del CR ResourceBackup que creaste en el paso 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (*Required*) Una cadena que indica qué acción realizará el gancho de ejecución, suponiendo que coincida cualquier filtro de gancho de ejecución especificado. Valores posibles:
     - Snapshot
     - Copia de seguridad
     - Restaurar
     - Conmutación por error
   - **spec.stage**: (*Required*) Una cadena que indica en qué etapa de la acción debe ejecutarse el gancho de ejecución. Esta ejecución de gancho no ejecutará ganchos en ninguna otra etapa. Valores posibles:
     - Pre
     - Publicar

     Ejemplo de 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. Después de rellenar el archivo CR con los valores correctos, aplica el CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Usa la CLI

Pasos

1. Crea la solicitud para ejecutar manualmente el gancho:

```
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. Comprueba el estado de la ejecución del hook de ejecución. Puedes ejecutar este comando repetidamente hasta que se complete la operación:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Describe el objeto exechooksrun para ver los detalles finales y el estado:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Desinstala Trident Protect

Es posible que tengas que eliminar componentes de Trident Protect si vas a actualizar de una versión de prueba a una versión completa del producto.

Para eliminar Trident Protect, realiza los siguientes pasos.

Pasos

1. Elimina los archivos Trident Protect CR:

   Este paso no es necesario para la versión 25.06 y posteriores.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Quita Trident Protect:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Elimina el espacio de nombres Trident Protect:

   ```
   kubectl delete ns trident-protect
   ```

Blogs de Trident y Trident Protect

Puedes encontrar algunos excelentes blogs de NetApp Trident y Trident Protect aquí:

Blogs de Trident

- 16 de octubre de 2025: "Soluciones avanzadas de almacenamiento para Kubernetes"
- 19 de agosto de 2025: "Mejorando la coherencia de los datos: instantáneas de grupos de volúmenes en OpenShift virtualización con Trident"
- 09 de mayo de 2025: "Configuración automática del backend Trident para FSx for ONTAP con el add-on Amazon EKS"
- 15 de abril de 2025: "NetApp Trident con Google Cloud NetApp Volumes para el protocolo SMB"
- 14 de abril de 2025: "Aprovechando el protocolo Fiber Channel con Trident 25.02 para almacenamiento persistente en Kubernetes"
- 14 de abril de 2025: "Desbloqueando el poder de los sistemas ASA r2 de NetApp para el almacenamiento en bloque de Kubernetes"
- 31 de marzo de 2025: "Simplificando la instalación de Trident en Red Hat OpenShift con el nuevo operador certificado"
- 27 de marzo de 2025: "Aprovisionamiento de Trident para SMB con Google Cloud NetApp Volumes"
- 05 de marzo de 2025: "Desbloquea la integración perfecta de almacenamiento iSCSI: una guía de FSxN en clústeres ROSA para AWS"
- 27 de febrero de 2025: "Implementar la identidad en la nube con Trident, GKE y Google Cloud NetApp Volumes"
- 12 de diciembre de 2024: "Presentamos la compatibilidad con Fibre Channel en Trident"
- 11 de noviembre de 2024: "NetApp Trident con Google Cloud NetApp Volumes"
- 29 de octubre de 2024: "Amazon FSx for NetApp ONTAP con Red Hat OpenShift Service on AWS (ROSA) usando Trident"
- 29 de octubre de 2024: "Migración en vivo de VMs con OpenShift Virtualization en ROSA y Amazon FSx for NetApp ONTAP"
- 08 de julio de 2024: "Uso de NVMe/TCP para consumir almacenamiento ONTAP para tus apps modernas en contenedores en Amazon EKS"
- 01 de julio de 2024: "Almacenamiento Kubernetes sin fisuras con Google Cloud NetApp Volumes Flex y Astra Trident"
- 11 de junio de 2024: "ONTAP como almacenamiento backend para el registro de imágenes integrado en OpenShift"

Blogs de Trident Protect

- 16 de mayo de 2025: "Automatizar la conmutación por error del registro para la recuperación ante desastres con los post-restore hooks de Trident Protect"
- 16 de mayo de 2025: "Recuperación de desastres de virtualización de OpenShift con NetApp Trident Protect"
- 13 de mayo de 2025: "Migración de storage class con Trident Protect backup restore"

- 09 de mayo de 2025: "Reescala aplicaciones de Kubernetes con los hooks post-restauración de Trident Protect"
- 03 de abril de 2025: "Trident Protect Power Up: replicación Kubernetes para protección y recuperación de desastres"
- 13 de marzo de 2025: "Operaciones de backup coherente ante fallos y restauración para VMs de OpenShift Virtualization"
- 11 de marzo de 2025: "Ampliando los patrones de GitOps a la protección de datos de aplicaciones con NetApp Trident"
- 03 mar 2025: "Trident 25.02: Elevando la experiencia de Red Hat OpenShift con nuevas y emocionantes funciones"
- 15 de enero de 2025: "Presentamos Trident Protect control de acceso basado en roles"
- 11 de noviembre de 2024: "Gestión de datos impulsada por Kubernetes: la nueva era con Trident Protect"

Conocimiento y soporte

Preguntas frecuentes

Encuentra respuestas a las preguntas más frecuentes sobre la instalación, configuración, actualización y solución de problemas de Trident.

Preguntas generales

¿Con qué frecuencia se publica Trident?

A partir de la versión 24.02, Trident se publica cada cuatro meses: febrero, junio y octubre.

¿Trident admite todas las características que se lanzan en una versión particular de Kubernetes?

Por lo general, Trident no admite funciones alfa en Kubernetes. Trident podría admitir funciones beta en las dos versiones de Trident que siguen a la versión beta de Kubernetes.

¿Tiene Trident alguna dependencia de otros productos de NetApp para su funcionamiento?

Trident no tiene dependencias de otros productos de software de NetApp y funciona como una aplicación independiente. Sin embargo, deberías tener un dispositivo de almacenamiento backend de NetApp.

¿Cómo puedo obtener todos los detalles de configuración de Trident?

Usa el comando `tridentctl get` para obtener más información sobre tu configuración de Trident.

¿Puedo obtener métricas sobre cómo se aprovisiona el almacenamiento por Trident?

Sí. Puntos finales de Prometheus que puedes usar para recopilar información sobre el funcionamiento de Trident, como el número de backends gestionados, el número de volúmenes aprovisionados, los bytes consumidos y así sucesivamente. También puedes usar "Cloud Insights" para monitoreo y análisis.

¿Cambia la experiencia del usuario al utilizar Trident como CSI Provisioner?

No. No hay cambios en lo que respecta a la experiencia del usuario y las funcionalidades. El nombre de provisioner utilizado es `csi.trident.netapp.io`. Este método de instalar Trident se recomienda si quieres usar todas las nuevas funciones que ofrecen las versiones actuales y futuras.

Instala y usa Trident en un clúster de Kubernetes

¿Trident admite una instalación sin conexión desde un registro privado?

Sí, Trident puede instalarse sin conexión. Consulta "Conoce la instalación de Trident".

¿Puedo instalar Trident de forma remota?

Sí. Trident 18.10 y versiones posteriores admiten la capacidad de instalación remota desde cualquier máquina que tenga `kubectl` acceso al clúster. Después de que se verifique el acceso `kubectl` (por ejemplo, inicia un comando `kubectl get nodes` desde la máquina remota para verificar), sigue las instrucciones de instalación.

¿Puedo configurar alta disponibilidad con Trident?

Trident se instala como un despliegue de Kubernetes (ReplicaSet) con una instancia, así que tiene HA incorporada. No deberías aumentar el número de réplicas en el despliegue. Si se pierde el nodo donde está instalado Trident o el pod no es accesible por cualquier motivo, Kubernetes vuelve a desplegar automáticamente el pod en un nodo sano de tu clúster. Trident es solo plano de control, así que los pods montados actualmente no se ven afectados si se vuelve a desplegar Trident.

¿Trident necesita acceso al espacio de nombres kube-system?

Trident lee del Kubernetes API Server para determinar cuándo las aplicaciones solicitan nuevos PVC, así que necesita acceso a kube-system.

¿Qué roles y privilegios utiliza Trident?

El instalador de Trident crea un ClusterRole de Kubernetes, que tiene acceso específico a los recursos PersistentVolume, PersistentVolumeClaim, StorageClass y Secret del clúster de Kubernetes. Consulta "Personaliza la instalación de tridentctl".

¿Puedo generar localmente los archivos de manifiesto exactos que Trident usa para la instalación?

Puedes generar y modificar localmente los archivos de manifiesto exactos que Trident usa para la instalación, si lo necesitas. Consulta "Personaliza la instalación de tridentctl".

¿Puedo compartir la misma SVM de backend de ONTAP para dos instancias de Trident separadas para dos clústeres de Kubernetes separados?

Aunque no es aconsejable, puedes usar el mismo backend SVM para dos instancias de Trident. Especifica un nombre de volumen único para cada instancia durante la instalación y/o especifica un parámetro único `StoragePrefix` en el archivo `setup/backend.json`. Esto es para asegurarte de que no se use el mismo volumen FlexVol para ambas instancias.

¿Es posible instalar Trident en ContainerLinux (formerly CoreOS)?

Trident es simplemente un pod de Kubernetes y puede instalarse donde Kubernetes esté en funcionamiento.

¿Puedo usar Trident con NetApp Cloud Volumes ONTAP?

Sí, Trident es compatible con AWS, Google Cloud y Azure.

Solución de problemas y soporte

¿NetApp es compatible con Trident?

Aunque Trident es de código abierto y se ofrece de forma gratuita, NetApp lo respalda completamente siempre que tu backend NetApp sea compatible.

¿Cómo abro un caso de soporte?

Para plantear un caso de asistencia, haz una de las siguientes acciones:

1. Ponte en contacto con tu Support Account Manager y consigue ayuda para crear un ticket.
2. Abre un caso de soporte contactando a "Soporte de NetApp".

¿Cómo genero un paquete de registro de soporte?

Puedes crear un paquete de soporte ejecutando `tridentctl logs -a`. Además de los registros capturados en el paquete, captura el registro de kubelet para diagnosticar los problemas de montaje en el lado de Kubernetes. Las instrucciones para obtener el registro de kubelet varían según cómo esté instalado Kubernetes.

¿Qué hago si necesito solicitar una nueva función?

Crea una incidencia en "Trident Github" y menciona **RFE** en el asunto y la descripción de la incidencia.

¿Dónde planteo un defecto?

Crea una incidencia en "Trident Github". Asegúrate de incluir toda la información necesaria y los registros relacionados con la incidencia.

¿Qué pasa si tengo una pregunta rápida sobre Trident que necesito aclarar? ¿Hay una comunidad o un foro?

Si tienes alguna pregunta, problema o solicitud, contáctanos a través de nuestro Trident "Canal de Discord" o GitHub.

La contraseña de mi sistema de almacenamiento ha cambiado y Trident ya no funciona, ¿cómo lo recupero?

Actualiza la contraseña del backend con `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Reemplaza `myBackend` en el ejemplo por el nombre de tu backend, y `` `/path/to_new_backend.json `` por la ruta al archivo correcto `backend.json`.

Trident no puede encontrar mi nodo Kubernetes. ¿Cómo arreglo esto?

Hay dos escenarios probables por los que Trident no puede encontrar un nodo Kubernetes. Puede ser por un problema de red dentro de Kubernetes o un problema de DNS. El daemonset de nodo de Trident que se ejecuta en cada nodo Kubernetes debe poder comunicarse con el controlador de Trident para registrar el nodo con Trident. Si hubo cambios en la red después de instalar Trident, solo tendrás este problema con los nuevos nodos Kubernetes que se agregan al clúster.

Si se destruye el pod de Trident, ¿perderé los datos?

Los datos no se perderán si se destruye el pod de Trident. Los metadatos de Trident se almacenan en objetos CRD. Todos los PV que han sido aprovisionados por Trident funcionarán normalmente.

Actualizar Trident

¿Puedo actualizar directamente de una versión anterior a una más nueva (saltándome algunas versiones)?

NetApp admite la actualización de Trident de una versión principal a la siguiente versión principal inmediata. Puedes actualizar de la versión 18.xx a la 19.xx, de la 19.xx a la 20.xx y así sucesivamente. Deberías probar la actualización en un laboratorio antes de la implementación en producción.

¿Es posible hacer downgrade de Trident a una versión anterior?

Si necesitas una solución para errores observados después de una actualización, problemas de dependencias

o una actualización fallida o incompleta, deberías "desinstala Trident" y reinstalar la versión anterior usando las instrucciones específicas para esa versión. Esta es la única forma recomendada de volver a una versión anterior.

Administra backends y volúmenes

¿Necesito definir tanto Management como DataLIF en un archivo de definición de backend de ONTAP?

El LIF de gestión es obligatorio. DataLIF varía:

- ONTAP SAN: no especifiques para iSCSI. Trident usa "Mapa LUN selectivo de ONTAP" para descubrir los LIF iSCI necesarios para establecer una sesión multirruta. Se genera una advertencia si `dataLIF` se define explícitamente. Consulta "Opciones de configuración y ejemplos de ONTAP SAN" para más detalles.
- ONTAP NAS: NetApp recomienda especificar `dataLIF`. Si no se proporciona, Trident obtiene los dataLIFs de la SVM. Puedes especificar un nombre de dominio completo (FQDN) para usar en las operaciones de montaje NFS, lo que te permite crear un DNS round-robin para balancear la carga entre varios dataLIFs. Consulta "Opciones de configuración y ejemplos de ONTAP NAS" para más detalles

¿Puede Trident configurar CHAP para backends ONTAP?

Sí. Trident admite CHAP bidireccional para backends ONTAP. Esto requiere establecer `useCHAP=true` en la configuración del backend.

¿Cómo gestiono las políticas de exportación con Trident?

Trident puede crear y gestionar dinámicamente políticas de exportación desde la versión 20.04 en adelante. Esto permite que el administrador de almacenamiento proporcione uno o más bloques CIDR en su configuración de backend y que Trident agregue las IP de los nodos que estén dentro de estos rangos a una política de exportación que crea. De esta manera, Trident gestiona automáticamente la adición y eliminación de reglas para los nodos con IP dentro de los CIDR dados.

¿Se pueden utilizar direcciones IPv6 para los Management y DataLIFs?

Trident admite definir direcciones IPv6 para:

- `managementLIF` and `dataLIF` para backends NAS de ONTAP.
- `managementLIF` para backends SAN de ONTAP. No puedes especificar `dataLIF` en un backend SAN de ONTAP.

Trident debe instalarse usando la bandera `--use-ipv6` (para `tridentctl` instalación), `IPv6` (para Trident operator) o `tridentTPv6` (para Helm installation) para que funcione sobre IPv6.

¿Es posible actualizar el LIF de gestión en el backend?

Sí, es posible actualizar el LIF de administración del backend usando el `tridentctl update backend` comando.

¿Es posible actualizar el DataLIF en el backend?

Puedes actualizar el DataLIF en `ontap-nas` y `ontap-nas-economy` solamente.

¿Puedo crear múltiples backends en Trident para Kubernetes?

Trident puede soportar muchos backends simultáneamente, ya sea con el mismo driver o con drivers diferentes.

¿Cómo almacena Trident las credenciales del backend?

Trident almacena las credenciales del backend como Kubernetes Secrets.

¿Cómo selecciona Trident un backend específico?

Si los atributos del backend no se pueden usar para seleccionar automáticamente los grupos correctos para una clase, los `storagePools` y `additionalStoragePools` parámetros se usan para seleccionar un conjunto específico de grupos.

¿Cómo me aseguro de que Trident no aprovisione desde un backend específico?

El `excludeStoragePools`parámetro se usa para filtrar el conjunto de pools que Trident usa para el aprovisionamiento y eliminará cualquier pool que coincida.

Si hay varios backends del mismo tipo, ¿cómo elige Trident qué backend usar?

Si hay varios backends configurados del mismo tipo, Trident selecciona el backend adecuado según los parámetros presentes en `StorageClass` y `PersistentVolumeClaim`. Por ejemplo, si hay varios backends del controlador ontap-nas, Trident intenta combinar los parámetros de `StorageClass` y `PersistentVolumeClaim` y encontrar un backend que pueda cumplir los requisitos enumerados en `StorageClass` y `PersistentVolumeClaim`. Si hay varios backends que coinciden con la solicitud, Trident selecciona uno de ellos aleatoriamente.

¿Trident admite CHAP bidireccional con Element/SolidFire?

Sí.

¿Cómo implementa Trident Qtrees en un volumen ONTAP? ¿Cuántos Qtrees se pueden implementar en un solo volumen?

El `ontap-nas-economy` controlador crea hasta 200 Qtrees en el mismo FlexVol volume (configurable entre 50 y 300), 100,000 Qtrees por nodo del clúster y 2.4M por clúster. Cuando introduces un nuevo `PersistentVolumeClaim` que es atendido por el controlador economy, el controlador verifica si ya existe un FlexVol volume que pueda atender el nuevo Qtree. Si el FlexVol volume no existe que pueda atender el Qtree, se crea un nuevo FlexVol volume.

¿Cómo puedo configurar permisos de Unix para volúmenes aprovisionados en ONTAP NAS?

Puedes establecer permisos de Unix en el volumen aprovisionado por Trident configurando un parámetro en el archivo de definición de backend.

¿Cómo puedo configurar un conjunto explícito de opciones de montaje NFS de ONTAP mientras aprovisiono un volumen?

De forma predeterminada, Trident no establece ningún valor en las opciones de montaje con Kubernetes. Para especificar las opciones de montaje en la clase de almacenamiento de Kubernetes, sigue el ejemplo dado "aquí".

¿Cómo configuro los volúmenes aprovisionados en una política de exportación específica?

Para permitir que los hosts apropiados accedan a un volumen, usa el `exportPolicy` parámetro configurado en el archivo de definición de backend.

¿Cómo configuro el cifrado de volumen con Trident y ONTAP?

Puedes configurar el cifrado en el volumen proporcionado por Trident usando el parámetro de cifrado en el archivo de definición del backend. Para más información, consulta: "Cómo funciona Trident con NVE y NAE"

¿Cuál es la mejor manera de implementar QoS para ONTAP mediante Trident?

Usa `StorageClasses` para implementar QoS para ONTAP.

¿Cómo especifico el aprovisionamiento thin o thick a través de Trident?

Los controladores de ONTAP admiten tanto aprovisionamiento fino como grueso. Los controladores de ONTAP usan por defecto el aprovisionamiento fino. Si quieres aprovisionamiento grueso, deberías configurar el archivo de definición del backend o el `StorageClass`. Si ambos están configurados, `StorageClass` tiene prioridad. Configura lo siguiente para ONTAP:

1. En `StorageClass`, establece el atributo `provisioningType` como grueso.
2. En el archivo de definición de backend, habilita los volúmenes gruesos configurando `backend spaceReserve parameter` como volumen.

¿Cómo me aseguro de que los volúmenes que estoy usando no se eliminen incluso si borro accidentalmente el PVC?

La protección de PVC se habilita automáticamente en Kubernetes desde la versión 1.10.

¿Puedo hacer crecer los PVC NFS que fueron creados por Trident?

Sí. Puedes expandir una PVC que ha sido creada por Trident. Ten en cuenta que el crecimiento automático de volúmenes es una función de ONTAP que no se aplica a Trident.

¿Puedo importar un volumen mientras está en SnapMirror Data Protection (DP) o en modo offline?

La importación del volumen falla si el volumen externo está en modo DP o está fuera de línea. Recibes el siguiente mensaje de error:

```
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.
```

¿Cómo se traduce la cuota de recursos a un clúster NetApp?

La cuota de recursos de almacenamiento de Kubernetes debería funcionar siempre que el almacenamiento de NetApp tenga capacidad. Cuando el almacenamiento de NetApp no puede respetar la configuración de la cuota de Kubernetes por falta de capacidad, Trident intenta aprovisionar pero da error.

¿Puedo crear instantáneas de volumen usando Trident?

Sí. La creación de instantáneas de volúmenes bajo demanda y de volúmenes persistentes a partir de instantáneas es compatible con Trident. Para crear volúmenes persistentes a partir de instantáneas, asegúrate de que la función `VolumeSnapshotDataSource` feature gate esté habilitada.

¿Cuáles son los controladores que admiten las instantáneas de volumen de Trident?

A partir de hoy, la compatibilidad con instantáneas a petición está disponible para nuestros `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` y `azure-netapp-files` controladores backend.

¿Cómo hago una copia de seguridad instantánea de un volumen aprovisionado por Trident con ONTAP?

Esto está disponible en `ontap-nas`, `ontap-san` y `ontap-nas-flexgroup` controladores. También puedes especificar un `snapshotPolicy` para el `ontap-san-economy` driver en el nivel FlexVol.

Esto también está disponible en los `ontap-nas-economy` drivers pero en la granularidad a nivel de volumen FlexVol y no en la granularidad a nivel de qtree. Para habilitar la capacidad de hacer instantáneas de volúmenes aprovisionados por Trident, establece la opción de parámetro de backend `snapshotPolicy` en la política de instantáneas que quieras, tal como se define en el backend de ONTAP. Cualquier instantánea tomada por el controlador de almacenamiento no es conocida por Trident.

¿Puedo establecer un porcentaje de reserva de snapshot para un volumen aprovisionado a través de Trident?

Sí, puedes reservar un porcentaje específico de espacio en disco para almacenar las copias snapshot a través de Trident configurando el atributo `snapshotReserve` en el archivo de definición del backend. Si has configurado `snapshotPolicy` y `snapshotReserve` en el archivo de definición del backend, el porcentaje de reserva de snapshot se establece según el porcentaje `snapshotReserve` mencionado en el archivo del backend. Si el número de porcentaje `snapshotReserve` no se menciona, ONTAP por defecto toma el porcentaje de reserva de snapshot como 5. Si la opción `snapshotPolicy` está configurada como none, el porcentaje de reserva de snapshot se establece en 0.

¿Puedo acceder directamente al directorio de instantáneas de volumen y copiar archivos?

Sí, puedes acceder al directorio de instantáneas en el volumen aprovisionado por Trident configurando el parámetro `snapshotDir` en el archivo de definición del backend.

¿Puedo configurar SnapMirror para volúmenes a través de Trident?

Actualmente, SnapMirror tiene que configurarse externamente usando ONTAP CLI o OnCommand System Manager.

¿Cómo restauro los volúmenes persistentes a una snapshot específica de ONTAP?

Para restaurar un volumen a una instantánea de ONTAP, sigue estos pasos:

1. Pon en reposo el pod de aplicación que está usando el volumen persistente.
2. Revierte a la instantánea que necesitas usando la CLI de ONTAP o OnCommand System Manager.
3. Reinicia el pod de la aplicación.

¿Puede Trident aprovisionar volúmenes en SVMs que tienen configurado un Load-Sharing Mirror?

Se pueden crear réplicas de carga compartida para volúmenes raíz de SVMs que sirven datos por NFS. ONTAP actualiza automáticamente las réplicas de carga compartida para volúmenes que han sido creados por Trident. Esto puede provocar retrasos al montar volúmenes. Cuando se crean varios volúmenes usando Trident, el aprovisionamiento de un volumen depende de que ONTAP actualice la réplica de carga compartida.

¿Cómo puedo separar el uso de la clase de almacenamiento para cada cliente/tenant?

Kubernetes no permite clases de almacenamiento en espacios de nombres. Sin embargo, puedes usar Kubernetes para limitar el uso de una clase de almacenamiento específica por espacio de nombres mediante cuotas de recursos de almacenamiento, que son por espacio de nombres. Para denegar a un espacio de nombres específico el acceso a un almacenamiento específico, establece la cuota de recursos en 0 para esa clase de almacenamiento.

Solución de problemas

Usa los consejos que se ofrecen aquí para solucionar los problemas que puedas encontrar al instalar y usar Trident.

> Si necesitas ayuda con Trident, crea un paquete de soporte usando `tridentctl logs -a -n trident` y envíalo a NetApp Support.

Solución de problemas general

- Si el pod Trident no se pone en marcha correctamente (por ejemplo, cuando el pod Trident está atascado en la fase `ContainerCreating` con menos de dos contenedores listos), ejecutar `kubectl -n trident describe deployment trident` y `kubectl -n trident describe pod trident-**` puede darte información adicional. Obtener registros de kubelet (por ejemplo, usando `journalctl -xeu kubelet`) también puede ser útil.
- Si no hay suficiente información en los registros de Trident, puedes intentar habilitar el modo de depuración para Trident pasando la bandera `-d` al parámetro de instalación según tu opción de instalación.

  Luego confirma que la depuración está configurada usando `./tridentctl logs -n trident` y buscando `level=debug msg` en el registro.

  Instalado con Operator

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

  Esto reiniciará todos los pods Trident, lo que puede tardar varios segundos. Puedes comprobarlo observando la columna 'AGE' en la salida de `kubectl get pod -n trident`.

  Para Trident 20.07 y 20.10 usa `tprov` en lugar de `torc`.

  Instalado con Helm

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Instalado con tridentctl

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

- También puedes obtener registros de depuración para cada backend incluyendo `debugTraceFlags` en la definición de tu backend. Por ejemplo, incluye `debugTraceFlags: {"api":true, "method":true,}` para obtener llamadas a la API y recorridos de métodos en los registros de Trident. Los backends existentes pueden tener `debugTraceFlags` configurado con un `tridentctl backend update`.
- Cuando uses Red Hat Enterprise Linux CoreOS (RHCOS), asegúrate de que `iscsid` esté habilitado en los nodos trabajadores y se inicie por defecto. Esto se puede hacer usando OpenShift MachineConfigs o modificando las plantillas de ignition.
- Un problema común que podrías encontrar al usar Trident con "Azure NetApp Files" es cuando los secretos de tenant y client vienen de un registro de app con permisos insuficientes. Para ver una lista completa de los requisitos de Trident, consulta "Azure NetApp Files" configuration.
- Si tienes problemas al montar un PV en un contenedor, asegúrate de que `rpcbind` está instalado y en ejecución. Usa el gestor de paquetes necesario para el sistema operativo anfitrión y revisa si `rpcbind` está en ejecución. Puedes comprobar el estado del servicio `rpcbind` ejecutando un `systemctl status rpcbind` o su equivalente.
- Si un backend Trident informa que está en el estado `failed` a pesar de haber funcionado antes, probablemente se deba a que se cambiaron las credenciales SVM/admin asociadas con el backend. Actualizar la información del backend usando `tridentctl update backend` o reiniciar el pod de Trident solucionará este problema.
- Si tienes problemas de permisos al instalar Trident con Docker como el runtime del contenedor, intenta instalar Trident con la opción `--in cluster=false`. Esto no usará un pod de instalación y evitará los problemas de permisos que se ven por el usuario `trident-installer`.
- Usa `uninstall parameter <Uninstalling Trident>` para limpiar después de una ejecución fallida. Por defecto, el script no elimina los CRDs que ha creado Trident, así que es seguro desinstalar e instalar de nuevo incluso en un despliegue en ejecución.
- Si quieres bajar a una versión anterior de Trident, primero ejecuta el comando `tridentctl uninstall` para eliminar Trident. Descarga el "Versión de Trident" que quieras e instálalo usando el comando `tridentctl install`.
- Después de una instalación correcta, si un PVC está atascado en la `Pending` fase, ejecutar `kubectl describe pvc` puede darte información adicional sobre por qué Trident no pudo aprovisionar un PV para este PVC.

Despliegue fallido de Trident usando el operador

Si estás desplegando Trident usando el operador, el estado de `TridentOrchestrator` cambia de `Installing` a `Installed`. Si ves el estado `Failed` y el operador no puede recuperarse solo, deberías revisar los registros del operador ejecutando el siguiente comando:

```
tridentctl logs -l trident-operator
```

El rastreo de los registros del contenedor trident-operator puede señalar dónde se encuentra el problema. Por ejemplo, uno de estos problemas podría ser la incapacidad para extraer las imágenes de contenedor requeridas de los registros upstream en un entorno airgapped.

Para entender por qué la instalación de Trident no tuvo éxito, deberías echar un vistazo al `TridentOrchestrator` estado.

```
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'
```

Este error indica que ya existe un `TridentOrchestrator` que se utilizó para instalar Trident. Dado que cada clúster de Kubernetes solo puede tener una instancia de Trident, el operador se asegura de que en un momento dado solo exista un `TridentOrchestrator` activo que pueda crear.

Además, observar el estado de los pods de Trident puede indicar a menudo si algo no va bien.

```
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
```

Puedes ver claramente que los pods no pueden inicializarse completamente porque una o más imágenes de contenedor no se obtuvieron.

Para solucionar el problema, debes editar el `TridentOrchestrator` CR. O también puedes eliminar `TridentOrchestrator` y crear uno nuevo con la definición modificada y precisa.

Despliegue fallido de Trident usando `tridentctl`

Para averiguar qué salió mal, puedes volver a ejecutar el instalador usando el argumento `-d`, lo que activará el modo de depuración y te ayudará a entender cuál es el problema:

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

Después de solucionar el problema, puedes limpiar la instalación de la siguiente manera y luego ejecutar de nuevo el comando `tridentctl install`:

```
./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.
```

Eliminar por completo Trident y CRDs

Puedes eliminar por completo Trident y todos los CRD creados y los recursos personalizados asociados.

Esto no se puede deshacer. No hagas esto a menos que quieras una instalación completamente nueva de Trident. Para desinstalar Trident sin eliminar los CRD, consulta "Desinstala Trident".

Operador de Trident

Para desinstalar Trident y eliminar por completo los CRDs usando el operador Trident:

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

Helm

Para desinstalar Trident y eliminar completamente los CRDs usando Helm:

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

<code>tridentctl</code>

Para eliminar completamente los CRD después de desinstalar Trident usando `tridentctl`

```
tridentctl obliviate crd
```

Fallo de desagrupación de nodo NVMe con espacios de nombres de bloque sin procesar RWX o Kubernetes 1.26

Si estás ejecutando Kubernetes 1.26, puede fallar la desagrupación del nodo al usar NVMe/TCP con namespaces de bloque sin procesar RWX. Los siguientes escenarios ofrecen una solución para el fallo. O también puedes actualizar Kubernetes a 1.27.

Eliminé el espacio de nombres y el pod

Imagina un escenario donde tienes un espacio de nombres gestionado por Trident (volumen persistente NVMe) adjunto a un pod. Si eliminas el espacio de nombres directamente desde el backend de ONTAP, el proceso de unstaging se queda atascado después de que intentes eliminar el pod. Este escenario no afecta al clúster de Kubernetes ni a otras funciones.

Solución alternativa

Desmonta el volumen persistente (correspondiente a ese namespace) del nodo respectivo y elimínalo.

DataLIFs bloqueados

```
 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.
.Solución alternativa
Levanta los dataLIFS para restaurar la funcionalidad completa.
```

Asignación de namespace eliminada

```
 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.
.Solución alternativa
Agrega el `hostNQN` de nuevo al subsistema.
```

Los clientes de NFSv4.2 informan "argumento no válido" después de actualizar ONTAP cuando esperaban que "v4.2-xattrs" estuviera activado

Después de actualizar ONTAP, los clientes NFSv4.2 pueden mostrar errores de "argumento no válido" al intentar montar exportaciones NFSv4.2. Este problema ocurre cuando la opción `v4.2-xattrs` no está habilitada en la SVM. Solución: habilita la opción `v4.2-xattrs` en la SVM o actualiza a ONTAP 9.12.1 o una versión posterior, donde esta opción ya viene habilitada por defecto.

Soporte

NetApp proporciona soporte para Trident de varias maneras. Hay amplias opciones gratuitas de auto-soporte disponibles 24x7, como artículos de la knowledgebase (KB) y un canal de Discord.

Ciclo de vida de soporte de Trident

Trident proporciona tres niveles de soporte según tu versión. Consulta "NetApp compatibilidad de versiones de software para definiciones".

Soporte completo

Trident ofrece soporte completo durante doce meses desde la fecha de lanzamiento.

Soporte limitado

Trident ofrece soporte limitado durante los meses 13 a 24 desde la fecha de lanzamiento.

Autoservicio

La documentación de Trident está disponible para los meses 25 - 36 desde la fecha de lanzamiento.

Versión	Soporte completo	Soporte limitado	Autoservicio

"25,10"	octubre de 2026	octubre de 2027	octubre de 2028
"25,06"	junio de 2026	junio de 2027	junio de 2028
"25,02"	febrero de 2026	febrero de 2027	febrero de 2028
"24,10"	—	octubre de 2026	octubre de 2027
"24,06"	—	junio de 2026	junio de 2027
"24,02"	—	febrero de 2026	febrero de 2027
"23,10"	—	—	octubre de 2026
"23,07"	—	—	julio de 2026
"23,04"	—	—	abril de 2026
"23,01"	—	—	enero de 2026

Autoservicio

Para ver una lista completa de artículos sobre solución de problemas, consulta "NetApp Knowledgebase (inicio de sesión obligatorio)".

Soporte de la comunidad

Hay una vibrante comunidad pública de usuarios de contenedores (incluidos los desarrolladores de Trident) en nuestro "Canal de Discord". Este es un gran lugar para hacer preguntas generales sobre el proyecto y hablar de temas relacionados con compañeros de ideas afines.

NetApp soporte técnico de NetApp

Para obtener ayuda con Trident, crea un paquete de soporte usando `tridentctl logs -a -n trident` y envíalo a `NetApp Support <Getting Help>`.

Para más información

- "Recursos de Trident"
- "Kubernetes Hub"

Referencia

Puertos de Trident

Conoce más sobre los puertos que usa Trident para la comunicación.

Descripción general

Trident utiliza varios puertos para la comunicación dentro de clústeres de Kubernetes y con storage backends. A continuación tienes un resumen de los puertos clave, sus propósitos y consideraciones de seguridad.

- **Enfoque de salida**: los nodos Kubernetes (controlador y trabajador) inician principalmente el tráfico hacia las LIF/IP de almacenamiento, así que las reglas de iptables deben permitir la salida desde las IP de los nodos hacia IP de almacenamiento específicas en estos puertos. Evita las reglas amplias "any-to-any".
- **Restricciones de entrada**: Limita los puertos internos de Trident al tráfico interno del clúster (por ejemplo, usando CNI como Calico). Sin exposición entrante innecesaria en los cortafuegos del host.
- **Seguridad de protocolo**:
  - Usa TCP siempre que sea posible (más fiable).
  - Habilita CHAP/IPsec para iSCSI si es sensible; TLS/HTTPS para gestión (puerto 443/8443).
  - Para NFSv4 (por defecto en Trident), elimina los puertos UDP/antiguos de NFSv3 (por ejemplo, 4045-4049) si no los necesitas.
  - Restringe a subredes de confianza; monitorea con herramientas como Prometheus (puerto 8001 opcional).

Puertos para nodos controladora

Estos puertos son principalmente para el operador de Trident (gestión de backend). Todos los puertos internos son de nivel pod; permite en los nodos solo si el firewall del host interfiere con CNI.

Puerto/Protocolo	Dirección	Propósito	Controlador/Protocolo	Notas de seguridad
TCP 8000	Entrada/Salida (cluster-internal)	Servidor REST de Trident (comunicaciones operator-controller)	Todo	Restringe a los CIDR de los pods; sin exposición externa.
TCP 8443	Entrada/Salida (cluster-internal)	Backchannel HTTPS (API interna segura)	Todo	Cifrado TLS; limita a la malla de servicios de Kubernetes si se usa.
TCP 8001	Entrada (clúster interno, opcional)	Métricas de Prometheus	Todo	Expón solo a herramientas de supervisión (por ejemplo, usando RBAC); desactiva si no se usa.
TCP 443	Salida	HTTPS a ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Requiere validación de certificado TLS; restringe solo a las IPs de mgmt LIF.

Puerto/Protocolo	Dirección	Propósito	Controlador/Protocolo	Notas de seguridad
TCP 8443	Salida	HTTPS a E-Series Web Services Proxy	E-Series (iSCSI)	API de REST por defecto; usa certificados; configurable en backend YAML.

Puertos para nodos worker

Estos puertos son para daemonsets de nodos CSI y montajes de pods. Los puertos de datos son salientes hacia los LIF de datos de almacenamiento; incluye extras de NFSv3 si usas NFSv3 (opcional para NFSv4).

Puerto/Protocolo	Dirección	Propósito	Controlador/Protocolo	Notas de seguridad
TCP 17546	Entrada (local a pod)	Sondas de liveness/readiness del nodo CSI	Todo	Configurable (--probe-port); asegúrate de que no haya conflictos de host; solo local.
TCP 8000	Entrada/Salida (cluster-internal)	Servidor REST de Trident	Todo	Como arriba; pod-interno.
TCP 8443	Entrada/Salida (cluster-internal)	HTTPS de canal secundario	Todo	Como arriba.
TCP 8001	Entrada (clúster interno, opcional)	Métricas de Prometheus	Todo	Como arriba.
TCP 443	Salida	HTTPS a ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Como en el caso anterior; se usa para el descubrimiento.
TCP 8443	Salida	HTTPS a E-Series Web Services Proxy	E-Series (iSCSI)	Como arriba.
TCP/UDP 111	Salida	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Obligatorio para v3; opcional para v4 (firewall offload); restringir si usas solo NFSv4.
TCP/UDP 2049	Salida	Demonio NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Datos básicos; bien conocidos; usa TCP para fiabilidad.
TCP/UDP 635	Salida	Daemon de montaje	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montaje; devoluciones de llamada bidireccionales posibles (permitir entrante efímero si es necesario).
UDP 4045	Salida	Gestor de bloqueos NFS (nlockmgr)	ONTAP-NAS (NFSv3)	Bloqueo de archivos; saltar para v4 (pNFS handles); UDP-only.

Puerto/Protocolo	Dirección	Propósito	Controlador/ Protocolo	Notas de seguridad
UDP 4046	Salida	Monitor de estado NFS (statd)	ONTAP-NAS (NFSv3)	Notificaciones; puede necesitar puertos efímeros de entrada (1024-65535) para callbacks.
UDP 4049	Salida	Demonio de cuotas NFS (rquotad)	ONTAP-NAS (NFSv3)	Cuotas; omitir para v4.
TCP 3260	Salida	objetivo iSCSI (discovery/data/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bien conocido; CHAP auth sobre este puerto; habilita CHAP mutuo para seguridad.
TCP 445	Salida	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bien conocido; usa SMB3 con cifrado (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (opcional)	Salida	Autenticación Kerberos	ONTAP (NFS/SMB/iSCSI con Kerb)	Si usas Kerberos (no por defecto); a servidores AD, no almacenamiento.
TCP/UDP 389 (opcional)	Salida	LDAP	ONTAP (NFS/SMB con LDAP)	Similar; para resolución de nombres/auth; restringir a AD.

El puerto de la sonda de actividad/preparación se puede cambiar durante la instalación usando la `--probe-port` bandera. Es importante asegurarte de que este puerto no esté siendo usado por otro proceso en los nodos de trabajo.

API de REST de Trident

Si bien "Comandos y opciones de tridentctl" son la forma más sencilla de interactuar con la API de REST de Trident, puedes usar el endpoint REST directamente si lo prefieres.

Cuándo usar la API de REST

La API de REST es útil para instalaciones avanzadas que usan Trident como un binario independiente en implementaciones que no son Kubernetes.

Para mayor seguridad, Trident `REST API` está restringido al host local por defecto cuando se ejecuta dentro de un pod. Para cambiar este comportamiento, necesitas establecer el argumento de Trident `-address` en la configuración de su pod.

Uso de la API de REST

Para ver ejemplos de cómo se llaman estas APIs, pasa el indicador de depuración (`-d`. Para más información, consulta "Administra Trident usando tridentctl".

La API funciona de la siguiente manera:

GET

`GET <trident-address>/trident/v1/<object-type>`

Enumera todos los objetos de ese tipo.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

Obtiene los detalles del objeto nombrado.

POST

`POST <trident-address>/trident/v1/<object-type>`

Crea un objeto del tipo especificado.

- Requiere una configuración JSON para el objeto que se va a crear. Para la especificación de cada tipo de objeto, consulta "Administra Trident usando tridentctl".
- Si el objeto ya existe, el comportamiento varía: los backends actualizan el objeto existente, mientras que todos los demás tipos de objetos fallarán la operación.

ELIMINAR

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Elimina el recurso nombrado.

> Los volúmenes asociados a backends o clases de almacenamiento seguirán existiendo; estos deben eliminarse por separado. Para más información, consulta "Administra Trident usando tridentctl".

Opciones de línea de comandos

Trident ofrece varias opciones de línea de comandos para el orquestador de Trident. Puedes usar estas opciones para modificar tu implementación.

Registro

`-debug`

Habilita la salida de depuración.

`-loglevel <level>`

Establece el nivel de registro (debug, info, warn, error, fatal). Por defecto es info.

Kubernetes

`-k8s_pod`

Usa esta opción o `-k8s_api_server` para habilitar la compatibilidad con Kubernetes. Al configurar esto, Trident usa las credenciales de la cuenta de servicio de Kubernetes del pod que lo contiene para contactar el servidor de API. Esto solo funciona cuando Trident se ejecuta como un pod en un clúster de Kubernetes con cuentas de servicio habilitadas.

-k8s_api_server <insecure-address:insecure-port>

Usa esta opción o `-k8s_pod` para habilitar el soporte para Kubernetes. Cuando se especifica, Trident se conecta al servidor de la API de Kubernetes usando la dirección y el puerto inseguros proporcionados. Esto permite que Trident se implemente fuera de un pod; sin embargo, solo admite conexiones inseguras al servidor de la API. Para conectarte de forma segura, implementa Trident en un pod con la opción `-k8s_pod`.

Docker

-volume_driver <name>

Nombre del controlador usado al registrar el complemento de Docker. Predeterminado a `netapp`.

-driver_port <port-number>

Escucha en este puerto en lugar de un socket de dominio UNIX.

-config <file>

Obligatorio; tienes que especificar esta ruta a un archivo de configuración de backend.

REST

-address <ip-or-host>

Especifica la dirección en la que el servidor REST de Trident debe escuchar. El valor predeterminado es localhost. Cuando escucha en localhost y se ejecuta dentro de un pod de Kubernetes, la interfaz REST no es accesible directamente desde fuera del pod. Usa `-address ""` para que la interfaz REST sea accesible desde la dirección IP del pod.

> La interfaz REST de Trident se puede configurar para escuchar y servir en 127.0.0.1 (para IPv4) o [::1] (para IPv6) solamente.

-port <port-number>

Especifica el puerto en el que debe escuchar el servidor REST de Trident. Por defecto es 8000.

-rest

Habilita la interfaz REST. El valor predeterminado es true.

Objetos de Kubernetes y Trident

Puedes interactuar con Kubernetes y Trident usando las API de REST leyendo y escribiendo objetos de recursos. Hay varios objetos de recursos que dictan la relación entre Kubernetes y Trident, Trident y el almacenamiento, y Kubernetes y el almacenamiento. Algunos de estos objetos se gestionan a través de Kubernetes y los otros se gestionan a través de Trident.

¿Cómo interactúan los objetos entre sí?

Quizá la forma más sencilla de entender los objetos, para qué sirven y cómo interactúan, es seguir una única solicitud de almacenamiento de un usuario de Kubernetes:

1. Un usuario crea un `PersistentVolumeClaim` solicitando un nuevo `PersistentVolume` de un tamaño determinado desde un Kubernetes `StorageClass` que fue previamente configurado por el administrador.
2. Kubernetes `StorageClass` identifica a Trident como su proveedor e incluye parámetros que le dicen a Trident cómo aprovisionar un volumen para la clase solicitada.
3. Trident busca en su propia `StorageClass` con el mismo nombre que identifica la coincidencia `Backends` y `StoragePools` que puede usar para aprovisionar volúmenes para la clase.
4. Trident aprovisiona almacenamiento en un backend coincidente y crea dos objetos: un `PersistentVolume` en Kubernetes que le dice a Kubernetes cómo encontrar, montar y tratar el volumen, y un volumen en Trident que mantiene la relación entre el `PersistentVolume` y el almacenamiento real.
5. Kubernetes vincula el `PersistentVolumeClaim` al nuevo `PersistentVolume`. Los pods que incluyen el `PersistentVolumeClaim` montan ese PersistentVolume en cualquier host donde se ejecuten.
6. Un usuario crea un `VolumeSnapshot` de un PVC existente usando un `VolumeSnapshotClass` que apunta a Trident.
7. Trident identifica el volumen que está asociado al PVC y crea una instantánea del volumen en su backend. También crea un `VolumeSnapshotContent` que le indica a Kubernetes cómo identificar la instantánea.
8. Un usuario puede crear un `PersistentVolumeClaim` usando `VolumeSnapshot` como fuente.
9. Trident identifica la instantánea requerida y realiza el mismo conjunto de pasos involucrados en la creación de un `PersistentVolume` y un `Volume`.

Para más información sobre los objetos de Kubernetes, te recomendamos que leas la sección "Volúmenes persistentes" de la documentación de Kubernetes.

Kubernetes `PersistentVolumeClaim` objetos

Un objeto de Kubernetes `PersistentVolumeClaim` es una solicitud de almacenamiento hecha por un usuario de un clúster de Kubernetes.

Además de la especificación estándar, Trident permite a los usuarios especificar las siguientes anotaciones específicas de volumen si quieren anular los valores predeterminados que tú configuraste en el backend:

Anotación	Opción de volumen	Controladores compatibles
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san,ontap-san-economy
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	protocolo	cualquier
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

Anotación	Opción de volumen	Controladores compatibles
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

Si el PV creado tiene la `Delete` política de recuperación, Trident elimina tanto el PV como el volumen de respaldo cuando el PV se libera (o sea, cuando el usuario elimina el PVC). Si la acción de eliminación falla, Trident marca el PV como tal y reintenta la operación periódicamente hasta que tenga éxito o el PV se elimine manualmente. Si el PV usa la `Retain` política, Trident lo ignora y asume que el administrador lo limpiará de Kubernetes y del backend, permitiendo que el volumen se respalde o se inspeccione antes de eliminarlo. Ten en cuenta que eliminar el PV no hace que Trident elimine el volumen de respaldo. Deberías eliminarlo usando la API de REST (`tridentctl`).

Trident admite la creación de instantáneas de volumen usando la especificación CSI: puedes crear una instantánea de volumen y usarla como fuente de datos para clonar los PVC existentes. De esta manera, las copias de un momento específico de los PV se pueden exponer a Kubernetes en forma de instantáneas. Luego, las instantáneas se pueden usar para crear nuevos PV. Echa un vistazo a `On-Demand Volume Snapshots` para ver cómo funcionaría esto.

Trident también proporciona las `cloneFromPVC` y `splitOnClone` anotaciones para crear clones. Puedes usar estas anotaciones para clonar un PVC sin tener que usar la implementación de CSI.

Aquí tienes un ejemplo: si un usuario ya tiene un PVC llamado `mysql`, el usuario puede crear un nuevo PVC llamado `mysqlclone` usando la anotación, como `trident.netapp.io/cloneFromPVC: mysql`. Con esta anotación establecida, Trident clona el volumen correspondiente al PVC mysql, en vez de aprovisionar un volumen desde cero.

Ten en cuenta los siguientes puntos:

- NetApp recomienda clonar un volumen inactivo.
- Un PVC y su clon deben estar en el mismo namespace de Kubernetes y tener la misma storage class.
- Con los `ontap-nas` y `ontap-san` controladores, podría ser deseable establecer la anotación PVC `trident.netapp.io/splitOnClone` junto con `trident.netapp.io/cloneFromPVC`. Con `trident.netapp.io/splitOnClone` configurado en `true`, Trident divide el volumen clonado del volumen primario y así desacopla completamente el ciclo de vida del volumen clonado de su volumen primario, aunque se pierde algo de eficiencia de almacenamiento. No establecer `trident.netapp.io/splitOnClone` o configurarlo en `false` reduce el consumo de espacio en el backend, pero crea dependencias entre el volumen primario y el clon, de modo que el volumen primario no se puede eliminar a menos que primero se elimine el clon. Un escenario donde tiene sentido dividir el clon es al clonar un volumen de base de datos vacío, cuando se espera que el volumen y su clon diverjan mucho y no se beneficien de las eficiencias de almacenamiento que ofrece ONTAP.

El `sample-input` directorio contiene ejemplos de definiciones de PVC para usar con Trident. Consulta para una descripción completa de los parámetros y configuraciones asociadas con los volúmenes de Trident.

Kubernetes `PersistentVolume` objetos

Un objeto de Kubernetes `PersistentVolume` representa una parte de almacenamiento que está disponible para el clúster de Kubernetes. Tiene un ciclo de vida independiente del pod que lo usa.

> Trident crea `PersistentVolume` objetos y los registra automáticamente en el clúster de Kubernetes según los volúmenes que aprovisiona. No se espera que los gestiones tú mismo.

Cuando creas un PVC que hace referencia a un recurso basado en Trident `StorageClass`, Trident aprovisiona un volumen nuevo usando la clase de almacenamiento correspondiente y registra un PV nuevo para ese volumen. Al configurar el volumen aprovisionado y el PV correspondiente, Trident sigue las siguientes reglas:

- Trident genera un nombre de PV para Kubernetes y un nombre interno que utiliza para aprovisionar el almacenamiento. En ambos casos, se asegura de que los nombres sean únicos en su ámbito.
- El tamaño del volumen coincide lo más posible con el tamaño solicitado en el PVC, aunque puede redondearse a la cantidad asignable más cercana, según la plataforma.

Kubernetes `StorageClass` objetos

Los objetos de Kubernetes `StorageClass` se especifican por nombre en `PersistentVolumeClaims` para aprovisionar almacenamiento con un conjunto de propiedades. La propia clase de almacenamiento identifica el aprovisionador que se va a usar y define ese conjunto de propiedades en términos que el aprovisionador entiende.

Es uno de los dos objetos básicos que necesita crear y gestionar el administrador. El otro es el objeto backend de Trident.

Un objeto de Kubernetes `StorageClass` que utiliza Trident se ve así:

```
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
```

Estos parámetros son específicos de Trident y le dicen a Trident cómo aprovisionar volúmenes para la clase.

Los parámetros de la clase de almacenamiento son:

Atributo	Tipo	Requerido	Descripción
atributos	map[string]string	no	Consulta la sección de atributos a continuación

Atributo	Tipo	Requerido	Descripción
storagePools	map[string]StringList	no	Mapa de nombres de backend a listas de pools de almacenamiento dentro
additionalStoragePools	map[string]StringList	no	Mapa de nombres de backend a listas de pools de almacenamiento dentro
excludeStoragePools	map[string]StringList	no	Mapa de nombres de backend a listas de pools de almacenamiento dentro

Los atributos de almacenamiento y sus posibles valores se pueden clasificar en atributos de selección de pool de almacenamiento y atributos de Kubernetes.

Atributos de selección de storage pool

Estos parámetros determinan qué grupos de almacenamiento administrados por Trident se deben utilizar para aprovisionar volúmenes de un tipo dado.

Atributo	Tipo	Valores	Oferta	Solicitud	Con el apoyo de
medios[1]	cadena	hdd, híbrido, ssd	El pool contiene medios de este tipo; híbrido significa ambos	Tipo de medio especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	cadena	delgado, grueso	Pool admite este método de aprovisionamiento	Método de aprovisionamiento especificado	grueso: todo ontap; fino: todo ontap & solidfire-san
backendType	cadena	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertenece a este tipo de backend	Backend especificado	Todos los drivers
instantáneas	bool	verdadero, falso	El pool admite volúmenes con instantáneas	Volumen con instantáneas habilitadas	ontap-nas, ontap-san, solidfire-san

Atributo	Tipo	Valores	Oferta	Solicitud	Con el apoyo de
clones	bool	verdadero, falso	El pool admite la clonación de volúmenes	Volumen con clones habilitados	ontap-nas, ontap-san, solidfire-san
cifrado	bool	verdadero, falso	El pool admite volúmenes cifrados	Volumen con cifrado habilitado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	entero	entero positivo	El pool es capaz de garantizar IOPS en este rango	Volumen garantizó estas IOPS	solidfire-san

[1]: No compatible con los sistemas ONTAP Select

En la mayoría de los casos, los valores solicitados influyen directamente en el aprovisionamiento; por ejemplo, solicitar un aprovisionamiento grueso da como resultado un volumen con aprovisionamiento grueso. Sin embargo, un Element storage pool utiliza su mínimo y máximo de IOPS ofrecidos para establecer los valores de QoS, en lugar del valor solicitado. En este caso, el valor solicitado se utiliza solo para seleccionar el storage pool.

Idealmente, puedes usar `attributes` solo para modelar las cualidades del almacenamiento que necesitas para satisfacer las necesidades de una clase en particular. Trident descubre y selecciona automáticamente los pools de almacenamiento que coinciden con *todos* los `attributes` que especifiques.

Si te encuentras sin poder usar `attributes` para seleccionar automáticamente los grupos correctos para una clase, puedes usar los parámetros `storagePools` y `additionalStoragePools` para refinar aún más los grupos o incluso para seleccionar un conjunto específico de grupos.

Puedes usar el `storagePools` parámetro para restringir aún más el conjunto de pools que coincidan con cualquier `attributes` especificado. En otras palabras, Trident usa la intersección de los pools identificados por los parámetros `attributes` y `storagePools` para el aprovisionamiento. Puedes usar cualquiera de los parámetros por separado o ambos juntos.

Puedes usar el `additionalStoragePools` parámetro para ampliar el conjunto de pools que Trident usa para el aprovisionamiento, sin importar los pools seleccionados por los parámetros `attributes` y `storagePools`.

Puedes usar el `excludeStoragePools`parámetro para filtrar el conjunto de pools que Trident usa para el aprovisionamiento. Usar este parámetro elimina cualquier pool que coincida.

En los `storagePools` y `additionalStoragePools` parámetros, cada entrada tiene el formato `<backend>:<storagePoolList>`, donde `<storagePoolList>` es una lista separada por comas de grupos de almacenamiento para el backend especificado. Por ejemplo, un valor para `additionalStoragePools` podría verse como `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Estas listas aceptan valores regex tanto para el backend como para los valores de la lista. Puedes usar `tridentctl get backend` para obtener la lista de backends y sus pools.

Atributos de Kubernetes

Estos atributos no influyen en la selección de pools de almacenamiento/backends por parte de Trident durante el aprovisionamiento dinámico. En su lugar, estos atributos simplemente proporcionan parámetros compatibles con Kubernetes Persistent Volumes. Los nodos de trabajo son responsables de las operaciones de creación del sistema de archivos y podrían requerir utilidades del sistema de archivos, como xfsprogs.

Atributo	Tipo	Valores	Descripción	Controladores relevantes	Versión de Kubernetes
fsType	cadena	ext4, ext3, xfs	El tipo de sistema de archivos para volúmenes de bloques	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Todo
allowVolumeExpansion	booleano	verdadero, falso	Habilitar o deshabilitar el soporte para aumentar el tamaño del PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	cadena	Inmediato, WaitForFirstConsumer	Elige cuándo ocurre la vinculación de volúmenes y el aprovisionamiento dinámico	Todo	1.19 - 1.26

- El `fsType` parámetro se usa para controlar el tipo de sistema de archivos deseado para los LUN de SAN. Además, Kubernetes también usa la presencia de `fsType` en una storage class para indicar que existe un sistema de archivos. La propiedad del volumen se puede controlar usando el `fsGroup` security context de un pod solo si `fsType` está configurado. Consulta "Kubernetes: configura un contexto de seguridad para un pod o contenedor" para ver una descripción general sobre cómo configurar la propiedad del volumen usando el `fsGroup` context. Kubernetes aplicará el valor de `fsGroup` solo si:

  - `fsType` se establece en la clase de almacenamiento.
  - El modo de acceso de PVC es RWO.

  Para los controladores de almacenamiento NFS, ya existe un sistema de archivos como parte de la exportación NFS. Para usar `fsGroup` la clase de almacenamiento, todavía necesitas especificar un `fsType`. Puedes configurarlo en `nfs` o en cualquier valor que no sea nulo.

- Consulta "Ampliar volúmenes" para más detalles sobre la expansión del volumen.
- El paquete de instalación de Trident proporciona varias definiciones de clases de almacenamiento de ejemplo para usar con Trident en `sample-input/storage-class-*.yaml`. Al eliminar una clase de almacenamiento de Kubernetes, también se elimina la clase de almacenamiento de Trident correspondiente.

Objetos de Kubernetes `VolumeSnapshotClass`

Los objetos de Kubernetes `VolumeSnapshotClass` son análogos a `StorageClasses`. Ayudan a definir múltiples clases de almacenamiento y son referenciados por instantáneas de volumen para asociar la instantánea con la clase de instantánea requerida. Cada instantánea de volumen está asociada a una sola clase de instantánea de volumen.

A `VolumeSnapshotClass` debe ser definido por un administrador para crear instantáneas. Una clase de instantánea de volumen se crea con la siguiente definición:

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

El `driver` especifica a Kubernetes que las solicitudes de instantáneas de volumen de la clase `csi-snapclass` son gestionadas por Trident. El `deletionPolicy` especifica la acción que se debe tomar cuando se debe eliminar una instantánea. Cuando `deletionPolicy` se establece en `Delete`, los objetos de instantánea de volumen, así como la instantánea subyacente en el clúster de almacenamiento, se eliminan al borrar una instantánea. De forma alternativa, si se establece en `Retain`, `VolumeSnapshotContent` y la instantánea física se conservan.

Objetos de Kubernetes `VolumeSnapshot`

Un objeto de Kubernetes `VolumeSnapshot` es una solicitud para crear una instantánea de un volumen. Así

como una PVC representa una solicitud hecha por un usuario para un volumen, una instantánea de volumen es una solicitud hecha por un usuario para crear una instantánea de una PVC existente.

Cuando llega una solicitud de instantánea de volumen, Trident gestiona automáticamente la creación de la instantánea para el volumen en el backend y la expone creando un objeto único `VolumeSnapshotContent`. Puedes crear instantáneas a partir de PVC existentes y usar las instantáneas como DataSource al crear nuevas PVC.

> El ciclo de vida de un VolumeSnapshot es independiente del PVC de origen: una instantánea persiste incluso después de que se elimine el PVC de origen. Al eliminar un PVC que tiene instantáneas asociadas, Trident marca el volumen de respaldo de este PVC en estado **Eliminando**, pero no lo elimina por completo. El volumen se elimina cuando se eliminan todas las instantáneas asociadas.

Objetos de Kubernetes `VolumeSnapshotContent`

Un objeto de Kubernetes `VolumeSnapshotContent` representa una instantánea tomada de un volumen ya aprovisionado. Es análogo a un `PersistentVolume` y significa una instantánea aprovisionada en el clúster de almacenamiento. Similar a `PersistentVolumeClaim` y `PersistentVolume` objetos, cuando se crea una instantánea, el objeto `VolumeSnapshotContent` mantiene una asignación uno a uno con el objeto `VolumeSnapshot` que solicitó la creación de la instantánea.

El `VolumeSnapshotContent`objeto contiene detalles que identifican de forma única la instantánea, como el `snapshotHandle`. Este `snapshotHandle`es una combinación única del nombre del PV y el nombre del `VolumeSnapshotContent`objeto.

Cuando se recibe una solicitud de instantánea, Trident crea la instantánea en el backend. Después de que se crea la instantánea, Trident configura un `VolumeSnapshotContent` objeto y así expone la instantánea a la API de Kubernetes.

> Normalmente, no necesitas administrar el `VolumeSnapshotContent` objeto. Una excepción a esto es cuando quieres "importar una instantánea de volumen" crear uno fuera de Trident.

Objetos de Kubernetes `VolumeGroupSnapshotClass`

Los objetos de Kubernetes `VolumeGroupSnapshotClass` son análogos a `VolumeSnapshotClass`. Ayudan a definir múltiples clases de almacenamiento y las instantáneas de grupo de volúmenes los referencian para asociar la instantánea con la clase de instantánea requerida. Cada instantánea de grupo de volúmenes está asociada a una sola clase de instantánea de grupo de volúmenes.

A `VolumeGroupSnapshotClass` debe ser definido por un administrador para crear un grupo de instantáneas. Una clase de instantánea de grupo de volúmenes se crea con la siguiente definición:

```
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
```

El `driver` especifica a Kubernetes que las solicitudes para instantáneas de grupo de volúmenes de la clase `csi-group-snap-class` son gestionadas por Trident. El `deletionPolicy` especifica la acción que se debe tomar cuando se debe eliminar una instantánea de grupo. Cuando `deletionPolicy` se establece en `Delete`, los objetos de la instantánea del grupo de volúmenes, así como la instantánea subyacente en el clúster de almacenamiento, se eliminan al borrar una instantánea. De manera alternativa, si se establece en `Retain`, significa que `VolumeGroupSnapshotContent` y la instantánea física se conservan.

Objetos de Kubernetes `VolumeGroupSnapshot`

Un objeto de Kubernetes `VolumeGroupSnapshot` es una solicitud para crear una snapshot de varios volúmenes. Así como un PVC representa una solicitud hecha por un usuario para un volumen, una snapshot de grupo de volúmenes es una solicitud hecha por un usuario para crear una snapshot de un PVC existente.

Cuando llega una solicitud de instantánea de grupo de volúmenes, Trident gestiona automáticamente la creación de la instantánea de grupo para los volúmenes en el backend y expone la instantánea mediante la creación de un `VolumeGroupSnapshotContent` objeto único. Puedes crear instantáneas a partir de PVC existentes y usar las instantáneas como DataSource al crear nuevos PVC.

> El ciclo de vida de un VolumeGroupSnapshot es independiente del PVC de origen: una instantánea persiste incluso después de que se elimine el PVC de origen. Al eliminar un PVC que tiene instantáneas asociadas, Trident marca el volumen de respaldo para este PVC en estado **Deleting** pero no lo elimina por completo. La instantánea del grupo de volúmenes se elimina cuando se eliminan todas las instantáneas asociadas.

Kubernetes `VolumeGroupSnapshotContent` objetos

Un objeto de Kubernetes `VolumeGroupSnapshotContent` representa una instantánea de grupo tomada de un volumen ya aprovisionado. Es análogo a `PersistentVolume` y significa una instantánea aprovisionada en el clúster de almacenamiento. De forma similar a `PersistentVolumeClaim` y `PersistentVolume` objetos, cuando se crea una instantánea, el objeto `VolumeSnapshotContent` mantiene una correspondencia uno a uno con el objeto `VolumeSnapshot` que había solicitado la creación de la instantánea.

El `VolumeGroupSnapshotContent` objeto contiene detalles que identifican al grupo de instantáneas, como el `volumeGroupSnapshotHandle` y volumeSnapshotHandles individuales existentes en el sistema de almacenamiento.

Cuando llega una solicitud de instantánea, Trident crea la instantánea del grupo de volúmenes en el backend. Después de que se crea la instantánea del grupo de volúmenes, Trident configura un `VolumeGroupSnapshotContent` objeto y así expone la instantánea a la API de Kubernetes.

Objetos de Kubernetes `CustomResourceDefinition`

Los recursos personalizados de Kubernetes son puntos finales en la API de Kubernetes que define el administrador y que se usan para agrupar objetos similares. Kubernetes admite la creación de recursos personalizados para almacenar una colección de objetos. Puedes obtener estas definiciones de recursos ejecutando `kubectl get crds`.

Las definiciones de recursos personalizadas (CRDs) y sus metadatos de objetos asociados son almacenados por Kubernetes en su almacén de metadatos. Esto elimina la necesidad de un almacén independiente para Trident.

Trident utiliza `CustomResourceDefinition` objetos para preservar la identidad de los objetos de Trident, como los backends de Trident, las clases de almacenamiento de Trident y los volúmenes de Trident. Estos objetos son gestionados por Trident. Además, el marco de instantáneas de volumen de CSI introduce algunos CRD que son necesarios para definir instantáneas de volumen.

Los CRD son una construcción de Kubernetes. Los objetos de los recursos definidos arriba son creados por Trident. Como ejemplo sencillo, cuando se crea un backend usando `tridentctl`, se crea un objeto CRD correspondiente `tridentbackends` para que lo consuma Kubernetes.

Aquí tienes algunos puntos que debes tener en cuenta sobre los CRD de Trident:

- Cuando se instala Trident, se crea un conjunto de CRDs que pueden usarse como cualquier otro tipo de recurso.
- Al desinstalar Trident usando el comando `tridentctl uninstall`, se eliminan los pods de Trident pero los CRD creados no se limpian. Consulta "Desinstala Trident" para entender cómo se puede eliminar por completo Trident y volver a configurarlo desde cero.

Trident `StorageClass` objetos

Trident crea clases de almacenamiento coincidentes para objetos de Kubernetes `StorageClass` que especifican `csi.trident.netapp.io` en su campo de provisioner. El nombre de la clase de almacenamiento coincide con el del objeto de Kubernetes `StorageClass` que representa.

> Con Kubernetes, estos objetos se crean automáticamente cuando un Kubernetes `StorageClass` que usa Trident como provisionador es registrado.

Las clases de almacenamiento comprenden un conjunto de requisitos para los volúmenes. Trident compara estos requisitos con los atributos presentes en cada storage pool; si coinciden, ese storage pool es un objetivo válido para aprovisionar volúmenes usando esa storage class.

Puedes crear configuraciones de clases de almacenamiento para definir directamente las clases de almacenamiento usando la API de REST. Sin embargo, para las implementaciones de Kubernetes, esperamos que se creen al registrar nuevos objetos de Kubernetes `StorageClass`.

Objetos backend de Trident

Los backends representan los proveedores de almacenamiento sobre los que Trident aprovisiona volúmenes; una sola instancia de Trident puede gestionar cualquier número de backends.

Este es uno de los dos tipos de objeto que tú creas y gestionas. El otro es el objeto de Kubernetes `StorageClass`.

Para más información sobre cómo construir estos objetos, consulta "configuración de backends".

Objetos de Trident `StoragePool`

Los pools de almacenamiento representan las distintas ubicaciones disponibles para el aprovisionamiento en cada backend. Para ONTAP, estas corresponden a agregados en SVMs. Para NetApp HCI/SolidFire, estas corresponden a bandas de QoS especificadas por el administrador. Cada pool de almacenamiento tiene un conjunto de atributos de almacenamiento distintos, que definen sus características de rendimiento y de protección de datos.

A diferencia de los demás objetos aquí, los candidatos a pool de almacenamiento siempre se descubren y gestionan automáticamente.

Objetos de Trident `Volume`

Los volúmenes son la unidad básica de aprovisionamiento, y comprenden puntos finales de backend, como recursos compartidos NFS y LUN iSCSI y FC. En Kubernetes, estos corresponden directamente a `PersistentVolumes`. Cuando crees un volumen, asegúrate de que tenga una clase de almacenamiento, que determina dónde se puede aprovisionar ese volumen, junto con un tamaño.

- En Kubernetes, estos objetos se gestionan automáticamente. Puedes consultarlos para ver qué aprovisionó Trident.
- Al eliminar un PV con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un estado **Deleting**. Para que el volumen Trident se elimine, tienes que eliminar las instantáneas del volumen.

Una configuración de volumen define las propiedades que debe tener un volumen aprovisionado.

Atributo	Tipo	Requerido	Descripción
versión	cadena	no	Versión de la API de Trident ("1")
nombre	cadena	sí	Nombre del volumen a crear
storageClass	cadena	sí	Clase de almacenamiento que se usará al aprovisionar el volumen
tamaño	cadena	sí	Tamaño del volumen a aprovisionar en bytes
protocolo	cadena	no	Tipo de protocolo a usar; "archivo" o "bloque"
internalName	cadena	no	Nombre del objeto en el sistema de almacenamiento; generado por Trident

Atributo	Tipo	Requerido	Descripción
cloneSourceVolume	cadena	no	ontap (nas, san) & solidfire-*: Nombre del volumen desde el que clonar
splitOnClone	cadena	no	ontap (nas, san): separa el clon de su padre
snapshotPolicy	cadena	no	ontap-*: política de SnapVault a utilizar
snapshotReserve	cadena	no	ontap-*: Porcentaje de volumen reservado para instantáneas
exportPolicy	cadena	no	ontap-nas*: política de exportación a utilizar
snapshotDirectory	bool	no	ontap-nas*: si el directorio de instantáneas es visible
unixPermissions	cadena	no	ontap-nas*: permisos UNIX iniciales
blockSize	cadena	no	solidfire-*: Tamaño de bloque/sector
fileSystem	cadena	no	Tipo de sistema de archivos
skipRecoveryQueue	cadena	no	Durante la eliminación de volumen, omite la cola de recuperación en el almacenamiento y elimina el volumen inmediatamente.

Trident genera `internalName` al crear el volumen. Esto consiste en dos pasos. Primero, antepone el prefijo de almacenamiento (ya sea el predeterminado `trident` o el prefijo en la configuración del backend) al nombre del volumen, lo que da como resultado un nombre con la forma `<prefix>-<volume-name>`. Luego procede a sanear el nombre, reemplazando los caracteres no permitidos en el backend. Para los backends ONTAP, reemplaza los guiones por guiones bajos (así, el nombre interno se convierte en `<prefix>_<volume-name>`). Para los backends Element, reemplaza los guiones bajos por guiones.

Puedes usar configuraciones de volumen para aprovisionar volúmenes directamente usando la API de REST, pero en las implementaciones de Kubernetes esperamos que la mayoría de los usuarios usen el método estándar de Kubernetes `PersistentVolumeClaim`. Trident crea este objeto de volumen automáticamente como parte del proceso de aprovisionamiento.

Objetos de Trident `Snapshot`

Las instantáneas son una copia de un momento específico de volúmenes, que pueden usarse para aprovisionar nuevos volúmenes o restaurar el estado. En Kubernetes, estas corresponden directamente a objetos `VolumeSnapshotContent`. Cada instantánea está asociada a un volumen, que es la fuente de los datos para la instantánea.

Cada `Snapshot` objeto incluye las propiedades que se indican a continuación:

Atributo	Tipo	Requerido	Descripción
versión	Cadena	Sí	Versión de la API de Trident ("1")
nombre	Cadena	Sí	Nombre del objeto de snapshot Trident
internalName	Cadena	Sí	Nombre del objeto de snapshot Trident en el sistema de almacenamiento
volumeName	Cadena	Sí	Nombre del volumen persistente para el que se crea la instantánea
volumeInternalName	Cadena	Sí	Nombre del objeto de volumen Trident asociado en el sistema de almacenamiento

En Kubernetes, estos objetos se gestionan automáticamente. Puedes consultarlos para ver qué aprovisionó Trident.

Cuando se crea una solicitud de objeto de Kubernetes `VolumeSnapshot`, Trident funciona creando un objeto de snapshot en el sistema de almacenamiento de respaldo. El `internalName` de este objeto de snapshot se genera combinando el prefijo `snapshot-` con el `UID` del objeto `VolumeSnapshot` (por ejemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` y `volumeInternalName` se rellenan obteniendo los detalles del volumen de respaldo.

Trident `ResourceQuota` objeto

El daemonset de Trident consume una `system-node-critical` Priority Class, la Priority Class más alta disponible en Kubernetes, para asegurarse de que Trident pueda identificar y limpiar volúmenes durante el apagado correcto de nodos y permitir que los pods del daemonset de Trident se adelanten a cargas de trabajo con una prioridad más baja en clústeres donde hay mucha presión de recursos.

Para lograr esto, Trident emplea un `ResourceQuota` objeto para asegurar que se satisface una Clase de Prioridad "system-node-critical" en el daemonset de Trident. Antes del despliegue y la creación del daemonset, Trident busca el `ResourceQuota` objeto y, si no lo descubre, lo aplica.

Si necesitas más control sobre la Cuota de Recursos y la Clase de Prioridad por defecto, puedes generar un `custom.yaml` o configurar el objeto `ResourceQuota` usando el Helm chart.

El siguiente es un ejemplo de un objeto `ResourceQuota` que prioriza el daemonset Trident.

```
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
```

Para más información sobre las cuotas de recursos, consulta "Kubernetes: cuotas de recursos".

Limpia `ResourceQuota` si falla la instalación

En el caso poco frecuente de que la instalación falle después de que se cree el objeto `ResourceQuota`, primero intenta "desinstalando" y luego vuelve a instalar.

Si eso no funciona, elimina manualmente el objeto `ResourceQuota`.

Eliminar `ResourceQuota`

Si prefieres controlar tu propia asignación de recursos, puedes eliminar el objeto Trident `ResourceQuota` usando el comando:

```
kubectl delete quota trident-csi -n trident
```

Normas de seguridad de los pods (PSS) y Security Context Constraints (SCC)

Las normas de seguridad de pods (PSS) y las políticas de seguridad de pods (PSP) de Kubernetes definen los niveles de permiso y restringen el comportamiento de los pods. OpenShift Security Context Constraints (SCC) también definen la restricción de pods específica del OpenShift Kubernetes Engine. Para proporcionar esta personalización, Trident habilita ciertos permisos durante la instalación. Las siguientes secciones detallan los permisos establecidos por Trident.

PSS sustituye a Pod Security Policies (PSP). PSP quedó obsoleta en Kubernetes v1.21 y se eliminará en v1.25. Para más información, consulta "Kubernetes: seguridad".

Contexto de seguridad requerido de Kubernetes y campos relacionados

Permiso	Descripción
Privilegiado	CSI requiere que los puntos de montaje sean bidireccionales, lo que significa que el pod del nodo Trident debe ejecutar un contenedor con privilegios. Para más información, consulta "Kubernetes: propagación de montajes".
Red de host	Necesario para el demonio iSCSI. `iscsiadm` gestiona los montajes iSCSI y usa la red del host para comunicarse con el demonio iSCSI.
IPC del host	NFS utiliza la comunicación entre procesos (IPC) para comunicarse con el NFSD.
PID del host	Necesario para iniciar `rpc-statd` para NFS. Trident consulta los procesos del host para determinar si `rpc-statd` se está ejecutando antes de montar volúmenes NFS.

Capacidades	La `SYS_ADMIN` capacidad se proporciona como parte de las capacidades predeterminadas para contenedores con privilegios. Por ejemplo, Docker establece estas capacidades para contenedores con privilegios:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	El perfil Seccomp siempre está "Unconfined" en los contenedores con privilegios; por lo tanto, no se puede habilitar en Trident.
SELinux	En OpenShift, los contenedores con privilegios se ejecutan en el `spc_t` ("Super Privileged Container") domain, y los contenedores sin privilegios se ejecutan en el `container_t` domain. En `containerd`, con `container-selinux` instalado, todos los contenedores se ejecutan en el `spc_t` domain, lo que efectivamente desactiva SELinux. Por lo tanto, Trident no agrega `seLinuxOptions` a los contenedores.
DAC	Los contenedores con privilegios deben ejecutarse como root. Los contenedores sin privilegios se ejecutan como root para acceder a los sockets unix requeridos por CSI.

Estándares de seguridad de pods (PSS)

Etiqueta	Descripción	Predeterminado
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permite que el controlador Trident y los nodos se admitan en el espacio de nombres de instalación. No cambies la etiqueta del espacio de nombres.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

> Cambiar las etiquetas del espacio de nombres puede hacer que los pods no se programen, aparezca un "Error creating: …" o un "Warning: trident-csi-…". Si esto pasa, revisa si la etiqueta del espacio de nombres para `privileged` fue cambiada. Si es así, reinstala Trident.

Políticas de seguridad de pods (PSP)

Campo	Descripción	Predeterminado
`allowPrivilegeEscalation`	Los contenedores privilegiados deben permitir la escalada de privilegios.	`true`
`allowedCSIDrivers`	Trident no usa volúmenes efímeros CSI en línea.	Vacío
`allowedCapabilities`	Los contenedores Trident sin privilegios no requieren más capacidades que el conjunto predeterminado y los contenedores privilegiados reciben todas las capacidades posibles.	Vacío
`allowedFlexVolumes`	Trident no utiliza un "Controlador de FlexVolume", por lo tanto no están incluidos en la lista de volúmenes permitidos.	Vacío
`allowedHostPaths`	El pod del nodo Trident monta el sistema de archivos raíz del nodo, así que no hay ningún beneficio en configurar esta lista.	Vacío
`allowedProcMountTypes`	Trident no utiliza ningún `ProcMountTypes`.	Vacío
`allowedUnsafeSysctls`	Trident no requiere nada inseguro `sysctls`.	Vacío
`defaultAddCapabilities`	No es necesario agregar capacidades a los contenedores privilegiados.	Vacío
`defaultAllowPrivilegeEscalation`	Permitir la escalada de privilegios se maneja en cada pod Trident.	`false`
`forbiddenSysctls`	No `sysctls` se permiten.	Vacío
`fsGroup`	Los contenedores Trident se ejecutan como root.	`RunAsAny`

Campo	Descripción	Predeterminado
`hostIPC`	Montar volúmenes NFS requiere que el IPC del host se comunique con `nfsd`	`true`
`hostNetwork`	iscsiadm requiere que la red del host se comunique con el daemon de iSCSI.	`true`
`hostPID`	Se requiere el PID del host para comprobar si `rpc-statd` está en ejecución en el nodo.	`true`
`hostPorts`	Trident no usa ningún puerto de host.	Vacío
`privileged`	Los pods de nodo Trident deben ejecutar un contenedor privilegiado para montar volúmenes.	`true`
`readOnlyRootFilesystem`	Los pods de nodo Trident deben escribir en el sistema de archivos del nodo.	`false`
`requiredDropCapabilities`	Los pods de nodo Trident ejecutan un contenedor privilegiado y no pueden eliminar capacidades.	`none`
`runAsGroup`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`runAsUser`	Los contenedores Trident se ejecutan como root.	`runAsAny`
`runtimeClass`	Trident no usa `RuntimeClasses`.	Vacío
`seLinux`	Trident no configura `seLinuxOptions` porque actualmente hay diferencias en cómo los tiempos de ejecución de contenedores y las distribuciones de Kubernetes manejan SELinux.	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`volumes`	Los pods Trident requieren estos plugins de volumen.	`hostPath, projected, emptyDir`

Restricciones de contexto de seguridad (SCC)

Etiquetas	Descripción	Predeterminado
`allowHostDirVolumePlugin`	Los pods de nodo Trident montan el sistema de archivos raíz del nodo.	`true`

Etiquetas	Descripción	Predeterminado
`allowHostIPC`	Montar volúmenes NFS requiere que el IPC del host se comunique con `nfsd`.	`true`
`allowHostNetwork`	iscsiadm requiere que la red del host se comunique con el daemon de iSCSI.	`true`
`allowHostPID`	Se requiere el PID del host para comprobar si `rpc-statd` está en ejecución en el nodo.	`true`
`allowHostPorts`	Trident no usa ningún puerto de host.	`false`
`allowPrivilegeEscalation`	Los contenedores privilegiados deben permitir la escalada de privilegios.	`true`
`allowPrivilegedContainer`	Los pods de nodo Trident deben ejecutar un contenedor privilegiado para montar volúmenes.	`true`
`allowedUnsafeSysctls`	Trident no requiere nada inseguro `sysctls`.	`none`
`allowedCapabilities`	Los contenedores Trident sin privilegios no requieren más capacidades que el conjunto predeterminado y los contenedores privilegiados reciben todas las capacidades posibles.	Vacío
`defaultAddCapabilities`	No es necesario agregar capacidades a los contenedores privilegiados.	Vacío
`fsGroup`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`groups`	Este SCC es específico de Trident y está vinculado a su usuario.	Vacío
`readOnlyRootFilesystem`	Los pods de nodo Trident deben escribir en el sistema de archivos del nodo.	`false`
`requiredDropCapabilities`	Los pods de nodo Trident ejecutan un contenedor privilegiado y no pueden eliminar capacidades.	`none`
`runAsUser`	Los contenedores Trident se ejecutan como root.	`RunAsAny`

Etiquetas	Descripción	Predeterminado
`seLinuxContext`	Trident no configura `seLinuxOptions` porque actualmente hay diferencias en cómo los tiempos de ejecución de contenedores y las distribuciones de Kubernetes manejan SELinux.	Vacío
`seccompProfiles`	Los contenedores privilegiados siempre se ejecutan "Unconfined".	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`users`	Se proporciona una entrada para vincular este SCC al usuario Trident en el namespace Trident.	n/a
`volumes`	Los pods Trident requieren estos plugins de volumen.	`hostPath, downwardAPI, projected, emptyDir`

Avisos legales

Los avisos legales proporcionan acceso a declaraciones de derechos de autor, marcas registradas, patentes y más.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marcas registradas

NETAPP, el logotipo de NETAPP y las marcas que aparecen en la página de NetApp Trademarks son marcas comerciales de NetApp, Inc. Otros nombres de empresas y productos pueden ser marcas comerciales de sus respectivos propietarios.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

Una lista actual de patentes propiedad de NetApp se puede encontrar en:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidad

"https://www.netapp.com/company/legal/privacy-policy/"

Código abierto

Puedes consultar los derechos de autor y las licencias de terceros usadas en el software de NetApp para Trident en el archivo de avisos de cada versión en https://github.com/NetApp/trident/.

Información de copyright

Copyright © 2026 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.