Referencia

Trident

NetApp
July 01, 2026

Tabla de contenidos

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 obtener 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 obtener 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 el soporte para 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. Por defecto es `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.

Objetos de Kubernetes `PersistentVolumeClaim`

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.

Objetos de Kubernetes `PersistentVolume`

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.

Objetos de Kubernetes `StorageClass`

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 manera alternativa, si se establece en `Retain`, significa que `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 nuevos 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 para este PVC en estado **Deleting** 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 `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 `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.

Objetos de Kubernetes `VolumeGroupSnapshotContent`

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.

Objetos de Trident `StorageClass`

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 obtener 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`