Referencia

Trident

NetApp
January 15, 2026

Tabla de contenidos

Referencia

Puertos Trident

Obtén más información sobre los puertos que Trident utiliza para la comunicación.

Puertos Trident

Trident utiliza los siguientes puertos para la comunicación dentro de Kubernetes:

Puerto	Objetivo
8443	HTTPS de canal de retorno
8001	punto de conexión de métricas de Prometheus
8000	Servidor REST Trident
17546	Puerto de sonda de actividad/preparación utilizado por los pods del daemonset de Trident

El puerto de la sonda de actividad/preparación se puede cambiar durante la instalación utilizando el `--probe-port` bandera. Es importante asegurarse de que este puerto no esté siendo utilizado por otro proceso en los nodos de trabajo.

API REST de Trident

Mientras"Comandos y opciones de tridentctl" son la forma más sencilla de interactuar con la API REST de Trident ; si lo prefiere, puede utilizar el punto de conexión REST directamente.

Cuándo usar la API REST

La API REST es útil para instalaciones avanzadas que utilizan Trident como binario independiente en implementaciones que no son de Kubernetes.

Para mayor seguridad, el Trident `REST API` Por defecto, está restringido a localhost cuando se ejecuta dentro de un pod. Para cambiar este comportamiento, debes configurar Trident `-address` argumento en su configuración de pod.

Utilizando la API REST

Para ver ejemplos de cómo se llaman estas API, pasa el paquete de depuración.(`-d`) bandera. Para obtener más información, consulte"Gestiona Trident usando tridentctl" .

La API funciona de la siguiente manera:

CONSEGUIR

`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.

CORREO

`POST <trident-address>/trident/v1/<object-type>`

Crea un objeto del tipo especificado.

- Requiere una configuración JSON para que se cree el objeto. Para la especificación de cada tipo de objeto, consulte"Gestiona 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.

BORRAR

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Elimina el recurso especificado.

> Los volúmenes asociados con los sistemas backend o las clases de almacenamiento seguirán existiendo; estos deben eliminarse por separado. Para obtener más información, consulte"Gestiona Trident usando tridentctl" .

Opciones de línea de comandos

Trident expone varias opciones de línea de comandos para el orquestador Trident . Puedes utilizar estas opciones para modificar tu implementación.

Explotación florestal

`-debug`

Habilita la salida de depuración.

`-loglevel <level>`

Establece el nivel de registro (depuración, información, advertencia, error, fatal). Valor predeterminado: información.

Kubernetes

`-k8s_pod`

Utilice esta opción o `-k8s_api_server` para habilitar la compatibilidad con Kubernetes. Al configurar esto, Trident utilizará las credenciales de la cuenta de servicio de Kubernetes del pod que lo contiene para contactar con el servidor de la 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>

Utilice esta opción o `-k8s_pod` para habilitar la compatibilidad con Kubernetes. Cuando se especifica, Trident se conecta al servidor de la API de Kubernetes utilizando la dirección y el puerto no seguros proporcionados. Esto permite que Trident se implemente fuera de un pod; sin embargo, solo admite conexiones no seguras al servidor de la API. Para conectarse de forma segura, implemente Trident en un pod con el `-k8s_pod` opción.

Docker

-volume_driver <name>

Nombre del controlador utilizado al registrar el complemento de Docker. Valor predeterminado `netapp` .

-driver_port <port-number>

Escuche en este puerto en lugar de en un socket de dominio UNIX.

-config <file>

Obligatorio; debe especificar esta ruta a un archivo de configuración del backend.

DESCANSAR

-address <ip-or-host>

Especifica la dirección en la que debe escuchar el servidor REST de Trident. Por defecto, se utiliza localhost. Al escuchar en localhost y ejecutarse dentro de un pod de Kubernetes, la interfaz REST no es directamente accesible desde fuera del pod. Usar `-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 solo en 127.0.0.1 (para IPv4) o [::1] (para IPv6).

-port <port-number>

Especifica el puerto en el que debe escuchar el servidor REST de Trident. Valor predeterminado: 8000.

-rest

Habilita la interfaz REST. Valor predeterminado: verdadero.

Objetos de Kubernetes y Trident

Puedes interactuar con Kubernetes y Trident utilizando API REST para leer y escribir objetos de recursos. Existen 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 otros a través de Trident.

¿Cómo interactúan los objetos entre sí?

Quizás la forma más sencilla de comprender los objetos, para qué sirven y cómo interactúan, sea 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 de un Kubernetes `StorageClass` que fue configurado previamente por el administrador.
2. Kubernetes `StorageClass` identifica a Trident como su proveedor e incluye parámetros que le indican a Trident cómo aprovisionar un volumen para la clase solicitada.
3. Trident analiza su propia situación. `StorageClass` con el mismo nombre que identifica la coincidencia `Backends` y `StoragePools` que puede utilizar para aprovisionar volúmenes para la clase.
4. Trident proporciona almacenamiento en un backend coincidente y crea dos objetos: un `PersistentVolume` en Kubernetes que le indica 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 enlaza el `PersistentVolumeClaim` a lo nuevo `PersistentVolume` . Cápsulas que incluyen `PersistentVolumeClaim` Monta ese PersistentVolume en cualquier host en el que se ejecute.
6. Un usuario crea un `VolumeSnapshot` de un PVC existente, utilizando un `VolumeSnapshotClass` eso apunta a Trident.
7. Trident identifica el volumen asociado al PVC y crea una instantánea del volumen en su sistema 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 una `PersistentVolume` y un `Volume` .

Para obtener más información sobre los objetos de Kubernetes, le recomendamos encarecidamente que lea el "Volúmenes persistentes" sección de la documentación de Kubernetes.

Kubernetes `PersistentVolumeClaim` objetos

Kubernetes `PersistentVolumeClaim` El objeto es una solicitud de almacenamiento realizada por un usuario de un clúster de Kubernetes.

Además de la especificación estándar, Trident permite a los usuarios especificar las siguientes anotaciones específicas del volumen si desean anular los valores predeterminados que se establecen en la configuración del backend:

Anotación	Opción de volumen	Controladores compatibles
trident.netapp.io/sistema de archivos	sistema de archivos	ontap-san, solidfire-san, economía ontap-san
trident.netapp.io/cloneFromPVC	clonar volumen de origen	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economy
trident.netapp.io/splitOnClone	dividirEnClon	ontap-nas, ontap-san
trident.netapp.io/protocolo	protocolo	cualquier
trident.netapp.io/exportPolicy	Política de exportación	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup

Anotación	Opción de volumen	Controladores compatibles
trident.netapp.io/snapshotPolicy	política de instantáneas	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/directorio de instantáneas	directorio de instantáneas	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	permisos unix	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/tamaño de bloque	tamaño del bloque	solidfire-san

Si el PV creado tiene el `Delete` Según la política de recuperación, Trident elimina tanto el PV como el volumen de respaldo cuando se libera el PV (es decir, cuando el usuario elimina el PVC). Si la acción de eliminación falla, Trident marca el PV como tal y reintenta periódicamente la operación hasta que tenga éxito o el PV se elimine manualmente. Si el PV utiliza el `Retain` Según la política de Trident , esto se ignora y se asume que el administrador lo eliminará de Kubernetes y del backend, lo que permite realizar una copia de seguridad o inspeccionar el volumen antes de su eliminación. Tenga en cuenta que eliminar el PV no provoca que Trident elimine el volumen de respaldo. Debes eliminarlo usando la API REST.(`tridentctl`).

Trident admite la creación de instantáneas de volumen mediante la especificación CSI: puede crear una instantánea de volumen y usarla como fuente de datos para clonar PVC existentes. De esta forma, se pueden exponer a Kubernetes copias puntuales de los PV en forma de instantáneas. Las instantáneas se pueden utilizar posteriormente para crear nuevos PV. Echa un vistazo a `On-Demand Volume Snapshots` para ver cómo funcionaría esto.

Trident también proporciona `cloneFromPVC` y `splitOnClone` Anotaciones para la creación de clones. Puedes usar estas anotaciones para clonar un PVC sin tener que usar la implementación CSI.

Aquí hay un ejemplo: Si un usuario ya tiene un PVC llamado `mysql` , el usuario puede crear un nuevo PVC llamado `mysqlclone` mediante el uso de la anotación, como por ejemplo `trident.netapp.io/cloneFromPVC: mysql` . Con este conjunto de anotaciones, Trident clona el volumen correspondiente al PVC de MySQL, en lugar de aprovisionar un volumen desde cero.

Considere los siguientes puntos:

- NetApp recomienda clonar un volumen inactivo.
- Un PVC y su clon deben estar en el mismo espacio de nombres de Kubernetes y tener la misma clase de almacenamiento.
- Con el `ontap-nas` y `ontap-san` Para los controladores, podría ser conveniente configurar la anotación PVC. `trident.netapp.io/splitOnClone` en conjunto con `trident.netapp.io/cloneFromPVC` . Con `trident.netapp.io/splitOnClone` empezar a `true` Trident separa el volumen clonado del volumen original y, por lo tanto, desacopla completamente el ciclo de vida del volumen clonado de su original a costa de perder cierta eficiencia de almacenamiento. No se configura `trident.netapp.io/splitOnClone` o configurarlo para `false` Esto da como resultado un menor consumo de espacio en el backend, a costa de crear dependencias entre los volúmenes padre y clon, de modo que el volumen padre no se puede eliminar a menos que primero se elimine el clon. Un escenario donde tiene sentido dividir el clon es clonar un volumen de base de datos vacío donde 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` El directorio contiene ejemplos de definiciones de PVC para usar con Trident. Referirse a Para obtener una descripción completa de los parámetros y la configuración asociados con los volúmenes de Trident .

Kubernetes `PersistentVolume` objetos

Kubernetes `PersistentVolume` El objeto representa una porción de almacenamiento que se pone a disposición del clúster de Kubernetes. Tiene un ciclo de vida independiente del pod que lo utiliza.

> Trident crea `PersistentVolume` objetos y los registra automáticamente en el clúster de Kubernetes en función de los volúmenes que aprovisiona. No se espera que usted los gestione.

Cuando creas un PVC que hace referencia a un Trident basado en `StorageClass` Trident aprovisiona un nuevo volumen utilizando la clase de almacenamiento correspondiente y registra un nuevo PV 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, resulta tranquilizador que los nombres sean únicos en su ámbito.
- El tamaño del volumen coincide lo más fielmente posible con el tamaño solicitado en el PVC, aunque podría redondearse al alza a la cantidad asignable más cercana, dependiendo de la plataforma.

Kubernetes `StorageClass` objetos

Kubernetes `StorageClass` Los objetos se especifican por nombre en `PersistentVolumeClaims` para aprovisionar el almacenamiento con un conjunto de propiedades. La propia clase de almacenamiento identifica el aprovisionador que se utilizará y define ese conjunto de propiedades en términos que el aprovisionador entiende.

Es uno de los dos objetos básicos que debe crear y gestionar el administrador. El otro es el objeto backend de Trident .

Kubernetes `StorageClass` Un objeto que utiliza Trident tiene este aspecto:

```
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 e indican a Trident cómo aprovisionar volúmenes para la clase.

Los parámetros de la clase de almacenamiento son:

Atributo	Tipo	Requerido	Descripción
atributos	map[cadena]cadena	No	Consulte la sección de atributos a continuación.
Grupos de almacenamiento	map[string]StringList	No	Mapa de nombres de backend a listas de grupos de almacenamiento dentro de
Grupos de almacenamiento adicionales	map[string]StringList	No	Mapa de nombres de backend a listas de grupos de almacenamiento dentro de
excluirStoragePools	map[string]StringList	No	Mapa de nombres de backend a listas de grupos de almacenamiento dentro de

Los atributos de almacenamiento y sus posibles valores se pueden clasificar en atributos de selección de grupo de almacenamiento y atributos de Kubernetes.

atributos de selección del grupo de almacenamiento

Estos parámetros determinan qué pools de almacenamiento gestionados por Trident deben utilizarse para aprovisionar volúmenes de un tipo determinado.

Atributo	Tipo	Valores	Oferta	Pedido	Con el apoyo de
medios[1]	cadena	disco duro, híbrido, SSD	La piscina contiene medios de este tipo; híbrido significa ambos	Tipo de medio especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
tipo de aprovisionamiento	cadena	delgado, grueso	Pool admite este método de aprovisionamiento.	Método de aprovisionamiento especificado	Espeso: todo Ontap; fino: todo Ontap y Solidfire-san

Atributo	Tipo	Valores	Oferta	Pedido	Con el apoyo de
Tipo de backend	cadena	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Pool pertenece a este tipo de backend.	Backend especificado	Todos los conductores
instantáneas	bool	verdadero, falso	El pool admite volúmenes con instantáneas.	Volumen con instantáneas habilitadas	ontap-nas, ontap-san, solidfire-san, gcp-cvs
clones	bool	verdadero, falso	Pool admite la clonación de volúmenes.	Volumen con clones habilitado	ontap-nas, ontap-san, solidfire-san, gcp-cvs
cifrado	bool	verdadero, falso	Pool admite volúmenes cifrados	Volumen con cifrado habilitado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	entero positivo	Pool es capaz de garantizar IOPS en este rango.	El volumen garantizaba 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 grupo de almacenamiento de Element utiliza sus valores mínimos y máximos de IOPS ofrecidos para establecer los valores de QoS, en lugar del valor solicitado. En este caso, el valor solicitado se utiliza únicamente para seleccionar el grupo de almacenamiento.

Idealmente, puedes usar `attributes` por sí solo para modelar las cualidades del almacenamiento que necesita para satisfacer las necesidades de una clase en particular. Trident descubre y selecciona automáticamente los grupos de almacenamiento que coinciden con *todos* los requisitos. `attributes` que usted especifique.

Si no puede usar `attributes` Para seleccionar automáticamente los grupos adecuados para una clase, puede utilizar el `storagePools` y `additionalStoragePools` parámetros para refinar aún más los grupos o incluso para seleccionar un conjunto específico de grupos.

Puedes utilizar el `storagePools` parámetro para restringir aún más el conjunto de pools que coincidan con cualquier especificado `attributes` . En otras palabras, Trident utiliza la intersección de pools identificada por el `attributes` y `storagePools` Parámetros para el aprovisionamiento. Puede utilizar cualquiera de los parámetros por separado o ambos juntos.

Puedes utilizar el `additionalStoragePools` parámetro para ampliar el conjunto de pools que Trident utiliza para el aprovisionamiento, independientemente de los pools seleccionados por el `attributes` y `storagePools` parámetros.

Puedes utilizar el `excludeStoragePools` Parámetro para filtrar el conjunto de pools que Trident utiliza para el aprovisionamiento. El uso de este parámetro elimina cualquier grupo que coincida.

En el `storagePools` y `additionalStoragePools` parámetros, cada entrada toma la forma `<backend>:<storagePoolList>` , dónde `<storagePoolList>` es una lista de grupos de almacenamiento separados por comas 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 de expresiones regulares tanto para el backend como para los valores de la lista. Puedes utilizar `tridentctl get backend` para obtener la lista de backends y sus pools.

atributos de Kubernetes

Estos atributos no influyen en la selección de pools/backends de almacenamiento por parte de Trident durante el aprovisionamiento dinámico. En cambio, estos atributos simplemente proporcionan parámetros compatibles con los volúmenes persistentes de Kubernetes. Los nodos de trabajo son responsables de las operaciones de creación del sistema de archivos y pueden requerir utilidades del sistema de archivos, como xfsprogs.

Atributo	Tipo	Valores	Descripción	Factores 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
permitirExpansiónDeVolumen	booleano	verdadero, falso	Habilitar o deshabilitar la compatibilidad con el aumento del tamaño del PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
modo de enlace de volumen	cadena	Inmediato, Esperar al primer consumidor	Elija cuándo se produce el enlace de volúmenes y el aprovisionamiento dinámico	Todo	1,19 - 1,26

- El `fsType` Este parámetro se utiliza para controlar el tipo de sistema de archivos deseado para las LUN SAN. Además, Kubernetes también utiliza la presencia de `fsType` en una clase de almacenamiento para indicar que existe un sistema de archivos. La propiedad del volumen se puede controlar mediante el `fsGroup` contexto de seguridad de un pod solo si `fsType` está listo. Referirse a"Kubernetes: Configurar un contexto de seguridad para un pod o contenedor" Para obtener una descripción general sobre cómo configurar la propiedad del volumen utilizando `fsGroup` contexto. Kubernetes aplicará `fsGroup` valor solo si:

  - `fsType`se establece en la clase de almacenamiento.
  - El modo de acceso al 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 aún necesita especificar un `fsType` Puedes configurarlo para `nfs` o cualquier valor no nulo.

- Referirse a"Ampliar volúmenes" Para obtener más detalles sobre la ampliación de 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` . Eliminar una clase de almacenamiento de Kubernetes provoca que también se elimine la clase de almacenamiento de Trident correspondiente.

Kubernetes `VolumeSnapshotClass` objetos

Kubernetes `VolumeSnapshotClass` Los objetos son análogos a `StorageClasses` . Ayudan a definir múltiples clases de almacenamiento y las instantáneas de volumen las utilizan como referencia para asociar la instantánea con la clase de instantánea requerida. Cada instantánea de volumen está asociada a una única clase de instantánea de volumen.

A `VolumeSnapshotClass` Debe ser definido por un administrador para poder crear instantáneas. Se crea una clase de instantánea de volumen 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 `csi-snapclass` Las clases son gestionadas por Trident. El `deletionPolicy` Especifica la acción que se debe realizar cuando se deba eliminar una instantánea. Cuando `deletionPolicy` está configurado para `Delete` Cuando se elimina una instantánea, se eliminan tanto los objetos de instantánea de volumen como la instantánea subyacente en el clúster de almacenamiento. Alternativamente, configurándolo a `Retain` significa que `VolumeSnapshotContent` y se conserva la instantánea física.

Kubernetes `VolumeSnapshot` objetos

Kubernetes `VolumeSnapshot` El objeto es una solicitud para crear una instantánea de un volumen. Así como

un PVC representa una solicitud realizada por un usuario para obtener un volumen, una instantánea de volumen es una solicitud realizada por un usuario para crear una instantánea de un PVC existente.

Cuando se recibe 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 identificador único. `VolumeSnapshotContent` objeto. Puede crear instantáneas a partir de PVC existentes y utilizarlas como fuente de datos 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 de este PVC en estado **Eliminando**, pero no lo elimina por completo. El volumen se elimina cuando se borran todas las instantáneas asociadas.

Kubernetes `VolumeSnapshotContent` objetos

Kubernetes `VolumeSnapshotContent` El objeto 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, `VolumeSnapshotContent` el objeto mantiene una correspondencia uno a uno con el `VolumeSnapshot` objeto, que había solicitado la creación de la instantánea.

El `VolumeSnapshotContent` El objeto contiene detalles que identifican de forma única la instantánea, como por ejemplo: `snapshotHandle` . Este `snapshotHandle` es una combinación única del nombre del PV y el nombre del `VolumeSnapshotContent` objeto.

Cuando llega una solicitud de instantánea, Trident crea la instantánea en el backend. Después de crear la instantánea, Trident configura un `VolumeSnapshotContent` objeto y, por lo tanto, expone la instantánea a la API de Kubernetes.

> Normalmente, no necesitas gestionar el `VolumeSnapshotContent` objeto. Una excepción a esto es cuando quieres"Importar una instantánea de volumen" Creado fuera de Trident.

Kubernetes `VolumeGroupSnapshotClass` objetos

Kubernetes `VolumeGroupSnapshotClass` Los objetos son análogos a `VolumeSnapshotClass` . Ayudan a definir múltiples clases de almacenamiento y las instantáneas de grupos de volúmenes las utilizan para asociar la instantánea con la clase de instantánea requerida. Cada instantánea de grupo de volúmenes está asociada a una única clase de instantánea de grupo de volúmenes.

A `VolumeGroupSnapshotClass` Debe ser definido por un administrador para crear un grupo de instantáneas. Se crea una clase de instantánea de grupo de volúmenes 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 de instantáneas de grupos de volúmenes de `csi-group-snap-class` Las clases son gestionadas por Trident. El `deletionPolicy` Especifica la acción que se debe realizar cuando se deba eliminar una instantánea de grupo. Cuando `deletionPolicy` está configurado para `Delete` Cuando se elimina una instantánea, se eliminan tanto los objetos de instantánea del grupo de volúmenes como la instantánea subyacente en el clúster de almacenamiento. Alternativamente, configurándolo a `Retain` significa que `VolumeGroupSnapshotContent` y se conserva la instantánea física.

Kubernetes `VolumeGroupSnapshot` objetos

Kubernetes `VolumeGroupSnapshot` El objeto es una solicitud para crear una instantánea de múltiples volúmenes. Así como un PVC representa una solicitud realizada por un usuario para obtener un volumen, una instantánea de grupo de volúmenes es una solicitud realizada por un usuario para crear una instantánea de un PVC existente.

Cuando se recibe 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 la expone creando un identificador único. `VolumeGroupSnapshotContent` objeto. Puede crear instantáneas a partir de PVC existentes y utilizarlas como fuente de datos 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 de este PVC en estado **Eliminando**, pero no lo elimina por completo. La instantánea del grupo de volúmenes se elimina cuando se borran todas las instantáneas asociadas.

Kubernetes `VolumeGroupSnapshotContent` objetos

Kubernetes `VolumeGroupSnapshotContent` El objeto representa una instantánea de grupo 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, `VolumeSnapshotContent` el objeto mantiene una correspondencia uno a uno con el `VolumeSnapshot` objeto, que había solicitado la creación de la instantánea.

El `VolumeGroupSnapshotContent` El objeto contiene detalles que identifican el grupo de instantáneas, como por ejemplo: `volumeGroupSnapshotHandle` y los identificadores de instantáneas de volumen 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 crear la instantánea del grupo de volúmenes, Trident configura un `VolumeGroupSnapshotContent` objeto y, por lo tanto, expone la instantánea a la API de Kubernetes.

Kubernetes `CustomResourceDefinition` objetos

Los recursos personalizados de Kubernetes son puntos de conexión en la API de Kubernetes que son definidos por el administrador y se utilizan para agrupar objetos similares. Kubernetes admite la creación de recursos personalizados para almacenar una colección de objetos. Puede obtener estas definiciones de recursos ejecutando `kubectl get crds` .

Kubernetes almacena las definiciones de recursos personalizados (CRD) y sus metadatos de objetos asociados en su almacén de metadatos. Esto elimina la necesidad de una tienda separada para Trident.

Trident utiliza `CustomResourceDefinition` objetos para preservar la identidad de los objetos Trident , como backends de Trident , clases de almacenamiento de Trident y volúmenes de Trident . Estos objetos son gestionados por Trident. Además, el marco de instantáneas de volumen CSI introduce algunos CRD que son necesarios para definir las instantáneas de volumen.

Los CRD son una construcción de Kubernetes. Los objetos de los recursos definidos anteriormente son creados por Trident. Como ejemplo sencillo, cuando se crea un backend utilizando `tridentctl` , un correspondiente `tridentbackends` El objeto CRD se crea para su uso por Kubernetes.

Aquí hay algunos puntos a tener en cuenta sobre los CRD de Trident:

- Cuando se instala Trident , se crea un conjunto de CRD que se pueden usar como cualquier otro tipo de recurso.
- Al desinstalar Trident utilizando el `tridentctl uninstall` Al ejecutar el comando, los pods de Trident se eliminan, pero los CRD creados no se limpian. Referirse a"Desinstalar Trident" para comprender cómo se puede eliminar completamente Trident y reconfigurarlo desde cero.

Trident `StorageClass` objetos

Trident crea clases de almacenamiento coincidentes para Kubernetes. `StorageClass` objetos que especifican `csi.trident.netapp.io` en su campo de aprovisionamiento. El nombre de la clase de almacenamiento coincide con el de Kubernetes. `StorageClass` objeto que representa.

> Con Kubernetes, estos objetos se crean automáticamente cuando se inicia una red Kubernetes. `StorageClass` que utiliza Trident como proveedor está registrado.

Las clases de almacenamiento comprenden un conjunto de requisitos para los volúmenes. Trident compara estos requisitos con los atributos presentes en cada grupo de almacenamiento; si coinciden, ese grupo de almacenamiento es un destino válido para el aprovisionamiento de volúmenes utilizando esa clase de almacenamiento.

Puede crear configuraciones de clases de almacenamiento para definir directamente las clases de almacenamiento mediante la API REST. Sin embargo, para las implementaciones de Kubernetes, esperamos que se creen al registrar un nuevo Kubernetes. `StorageClass` objetos.

objetos de backend de Trident

Los backends representan los proveedores de almacenamiento sobre los cuales Trident aprovisiona volúmenes; una sola instancia de Trident puede administrar cualquier número de backends.

Este es uno de los dos tipos de objetos que usted crea y administra usted mismo. El otro es Kubernetes. `StorageClass` objeto.

Para obtener más información sobre cómo construir estos objetos, consulte"configuración de backends" .

Trident `StoragePool` objetos

Los grupos de almacenamiento representan las distintas ubicaciones disponibles para el aprovisionamiento en cada backend. Para ONTAP, estos corresponden a agregados en SVM. Para NetApp HCI/ SolidFire, estos corresponden a las bandas QoS especificadas por el administrador. Para el Cloud Volumes Service, estos corresponden a las regiones del proveedor de la nube. Cada grupo 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 presentes, los candidatos a grupo de almacenamiento siempre se descubren y gestionan automáticamente.

Trident `Volume` objetos

Los volúmenes son la unidad básica de aprovisionamiento, que comprende puntos de conexión de backend, como recursos compartidos NFS y LUN iSCSI y FC. En Kubernetes, estos se corresponden directamente con `PersistentVolumes` . Cuando cree un volumen, asegúrese 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 provisionó Trident .
- Al eliminar un PV con instantáneas asociadas, el volumen Trident correspondiente se actualiza al estado **Eliminando**. Para eliminar el volumen de Trident , debe eliminar las instantáneas del volumen.

La configuración de un 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
clase de almacenamiento	cadena	Sí	Clase de almacenamiento que se utilizará al aprovisionar el volumen
tamaño	cadena	Sí	Tamaño del volumen a aprovisionar en bytes
protocolo	cadena	No	Tipo de protocolo a utilizar: "archivo" o "bloque".

Atributo	Tipo	Requerido	Descripción
nombre interno	cadena	No	Nombre del objeto en el sistema de almacenamiento; generado por Trident
clonar volumen de origen	cadena	No	ontap (nas, san) y solidfire-*: Nombre del volumen a clonar
dividirEnClon	cadena	No	ontap (nas, san): Separa el clon de su padre
política de instantáneas	cadena	No	ontap-*: Política de instantánea a utilizar
snapshotReserve	cadena	No	ontap-*: Porcentaje de volumen reservado para instantáneas
Política de exportación	cadena	No	ontap-nas*: Política de exportación a utilizar
directorio de instantáneas	bool	No	ontap-nas*: Indica si el directorio de instantáneas es visible.
permisos unix	cadena	No	ontap-nas*: Permisos UNIX iniciales
tamaño del bloque	cadena	No	solidfire-*: Tamaño del bloque/sector
sistema de archivos	cadena	No	tipo de sistema de archivos

Trident genera `internalName` al crear el volumen. Esto consta de 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 de la forma `<prefix>-<volume-name>` . A continuación, procede a sanear el nombre, reemplazando los caracteres no permitidos en el backend. Para los backends de ONTAP , reemplaza los guiones con guiones bajos (por lo tanto, el nombre interno se convierte en `<prefix>_<volume-name>`). Para los backends de Element, reemplaza los guiones bajos con guiones.

Puedes usar configuraciones de volumen para aprovisionar volúmenes directamente mediante la API REST, pero en implementaciones de Kubernetes esperamos que la mayoría de los usuarios utilicen la configuración estándar de Kubernetes. `PersistentVolumeClaim` método. Trident crea este objeto de volumen automáticamente como parte del proceso de aprovisionamiento.

Trident `Snapshot` objetos

Las instantáneas son una copia puntual de los volúmenes, que se puede utilizar para aprovisionar nuevos volúmenes o restaurar el estado. En Kubernetes, estos se corresponden directamente con `VolumeSnapshotContent` objetos. Cada instantánea está asociada a un volumen, que es la fuente de los datos para la instantánea.

Cada `Snapshot` El objeto incluye las propiedades que se enumeran a continuación:

Atributo	Tipo	Requerido	Descripción
versión	Cadena	Sí	Versión de la API de Trident ("1")
nombre	Cadena	Sí	Nombre del objeto de instantánea de Trident
nombre interno	Cadena	Sí	Nombre del objeto de instantánea de Trident en el sistema de almacenamiento
nombre del volumen	Cadena	Sí	Nombre del volumen persistente para el que se crea la instantánea
nombre interno del volumen	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 provisionó Trident .

Cuando un Kubernetes `VolumeSnapshot` Se crea una solicitud de objeto; Trident funciona creando un objeto de instantánea en el sistema de almacenamiento de respaldo. El `internalName` Este objeto de instantánea se genera combinando el prefijo `snapshot-` con el `UID` del `VolumeSnapshot` objeto (por ejemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` y `volumeInternalName` se completan obteniendo los detalles del volumen de respaldo.

Trident `ResourceQuota` objeto

El conjunto de demonios Trident consume un `system-node-critical` Clase de prioridad: la clase de prioridad más alta disponible en Kubernetes, para garantizar que Trident pueda identificar y limpiar volúmenes durante el apagado ordenado de nodos y permitir que los pods del conjunto de demonios de Trident prevalezcan sobre las cargas de trabajo con una prioridad más baja en clústeres donde hay una alta presión de recursos.

Para lograrlo, Trident emplea un `ResourceQuota` objeto para garantizar que se cumpla una clase de prioridad "crítica del nodo del sistema" en el conjunto de demonios Trident . Antes del despliegue y la creación del DaemonSet, Trident busca el `ResourceQuota` objeto y, si no lo descubre, lo aplica.

Si necesita un mayor control sobre la cuota de recursos y la clase de prioridad predeterminadas, puede generar un `custom.yaml` o configurar el `ResourceQuota` objeto que utiliza el gráfico Helm.

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 obtener más información sobre las cuotas de recursos, consulte"Kubernetes: Cuotas de recursos" .

Limpiar `ResourceQuota` si falla la instalación

En el raro caso de que la instalación falle después de la `ResourceQuota` Se crea el objeto, primer intento"desinstalación" y luego reinstalar.

Si eso no funciona, elimine manualmente el `ResourceQuota` objeto.

Eliminar `ResourceQuota`

Si prefieres controlar tu propia asignación de recursos, puedes eliminar el Trident. `ResourceQuota` objeto usando el comando:

```
kubectl delete quota trident-csi -n trident
```

Estándares de seguridad de pods (PSS) y restricciones de contexto de seguridad (SCC)

Los estándares de seguridad de pods (PSS) y las políticas de seguridad de pods (PSP) de Kubernetes definen los niveles de permisos y restringen el comportamiento de los pods. De manera similar, las restricciones del contexto de seguridad (SCC) de OpenShift definen restricciones de pods específicas del motor de Kubernetes de OpenShift. Para ofrecer esta personalización, Trident habilita ciertos permisos durante la instalación. Las siguientes secciones detallan los permisos establecidos por Trident.

PSS reemplaza a las Políticas de Seguridad de Pod (PSP). PSP quedó obsoleto en Kubernetes v1.21 y se eliminará en la versión v1.25. Para obtener más información, consulte"Kubernetes: Seguridad" .

Contexto de seguridad de Kubernetes requerido 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 privilegiado. Para obtener más información, consulte"Kubernetes: Propagación de montaje" .
Redes de host	Requerido para el demonio iSCSI. `iscsiadm` Gestiona los montajes iSCSI y utiliza la red del host para comunicarse con el demonio iSCSI.
IPC del host	NFS utiliza comunicación entre procesos (IPC) para comunicarse con NFSD.
PID del host	Requisitos para comenzar `rpc-statd` para NFS. Trident consulta los procesos del host para determinar si `rpc-statd` se ejecuta antes de montar los volúmenes NFS.

Capacidades	El `SYS_ADMIN` Esta capacidad se proporciona como parte de las capacidades predeterminadas para contenedores privilegiados. Por ejemplo, Docker establece estas capacidades para los contenedores privilegiados:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	El perfil Seccomp siempre está "Sin restricciones" en contenedores privilegiados; por lo tanto, no se puede habilitar en Trident.
SELinux	En OpenShift, los contenedores privilegiados se ejecutan en el `spc_t` («Contenedor con privilegios superiores»), y los contenedores sin privilegios se ejecutan en el dominio … sin privilegios»). `container_t` dominio. En `containerd` , con `container-selinux` instalados, todos los contenedores se ejecutan en el `spc_t` dominio, lo que efectivamente deshabilita SELinux. Por lo tanto, Trident no agrega `seLinuxOptions` a los contenedores.
DAC	Los contenedores privilegiados 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	Por defecto
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permite que el controlador Trident y los nodos sean admitidos en el espacio de nombres de instalación. No cambie 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 provocar que los pods no se programen, un "Error al crear: …" o "Advertencia: trident-csi-…". Si esto ocurre, compruebe si la etiqueta del espacio de nombres para `privileged` fue cambiado En ese caso, reinstale Trident.

Políticas de seguridad de pods (PSP)

Campo	Descripción	Por defecto
`allowPrivilegeEscalation`	Los contenedores privilegiados deben permitir la escalada de privilegios.	`true`
`allowedCSIDrivers`	Trident no utiliza volúmenes efímeros CSI en línea.	Vacío
`allowedCapabilities`	Los contenedores Trident no privilegiados no requieren más capacidades que el conjunto predeterminado, y a los contenedores privilegiados se les otorgan todas las capacidades posibles.	Vacío
`allowedFlexVolumes`	Trident no utiliza un"Controlador 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, por lo tanto, no hay ningún beneficio en configurar esta lista.	Vacío
`allowedProcMountTypes`	Trident no utiliza ningún `ProcMountTypes` .	Vacío
`allowedUnsafeSysctls`	Trident no requiere ningún método inseguro. `sysctls` .	Vacío
`defaultAddCapabilities`	No es necesario agregar capacidades a los contenedores privilegiados.	Vacío
`defaultAllowPrivilegeEscalation`	La gestión de la escalada de privilegios se realiza en cada pod de Trident .	`false`
`forbiddenSysctls`	No `sysctls` están permitidos.	Vacío

Campo	Descripción	Por defecto
`fsGroup`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`hostIPC`	El montaje de 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 demonio iSCSI.	`true`
`hostPID`	Se requiere el PID del host para comprobar si `rpc-statd` se está ejecutando en el nodo.	`true`
`hostPorts`	Trident no utiliza ningún puerto del host.	Vacío
`privileged`	Los pods de nodos Trident deben ejecutar un contenedor con privilegios para poder montar volúmenes.	`true`
`readOnlyRootFilesystem`	Los pods de nodo Trident deben escribir en el sistema de archivos del nodo.	`false`
`requiredDropCapabilities`	Los pods de nodos Trident ejecutan un contenedor privilegiado y no pueden descartar capacidades.	`none`
`runAsGroup`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`runAsUser`	Los contenedores Trident se ejecutan como root.	`runAsAny`
`runtimeClass`	Trident no utiliza `RuntimeClasses` .	Vacío
`seLinux`	Trident no se configura `seLinuxOptions` porque actualmente existen diferencias en la forma en que los entornos de ejecución de contenedores y las distribuciones de Kubernetes gestionan SELinux.	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`volumes`	Los pods Trident requieren estos plugins de volumen.	`hostPath, projected, emptyDir`

Restricciones del contexto de seguridad (SCC)

Etiquetas	Descripción	Por defecto
`allowHostDirVolumePlugin`	Los pods de nodo Trident montan el sistema de archivos raíz del nodo.	`true`
`allowHostIPC`	El montaje de 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 demonio iSCSI.	`true`
`allowHostPID`	Se requiere el PID del host para comprobar si `rpc-statd` se está ejecutando en el nodo.	`true`
`allowHostPorts`	Trident no utiliza ningún puerto del host.	`false`
`allowPrivilegeEscalation`	Los contenedores privilegiados deben permitir la escalada de privilegios.	`true`
`allowPrivilegedContainer`	Los pods de nodos Trident deben ejecutar un contenedor con privilegios para poder montar volúmenes.	`true`
`allowedUnsafeSysctls`	Trident no requiere ningún método inseguro. `sysctls` .	`none`
`allowedCapabilities`	Los contenedores Trident no privilegiados no requieren más capacidades que el conjunto predeterminado, y a los contenedores privilegiados se les otorgan 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 nodos Trident ejecutan un contenedor privilegiado y no pueden descartar capacidades.	`none`

Etiquetas	Descripción	Por defecto
`runAsUser`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`seLinuxContext`	Trident no se configura `seLinuxOptions` porque actualmente existen diferencias en la forma en que los entornos de ejecución de contenedores y las distribuciones de Kubernetes gestionan SELinux.	Vacío
`seccompProfiles`	Los contenedores privilegiados siempre se ejecutan "sin restricciones".	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como root.	`RunAsAny`
`users`	Se proporciona una entrada para vincular este SCC al usuario Trident en el espacio de nombres Trident .	n / A
`volumes`	Los pods Trident requieren estos plugins de volumen.	`hostPath, downwardAPI, projected, emptyDir`