Referencia

Astra Trident

NetApp
September 04, 2024

Tabla de contenidos

Referencia

Puertos Astra Trident

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

Puertos Astra Trident

Astra Trident se comunica mediante los siguientes puertos:

Puerto	Específico
8443	HTTPS de canal posterior
8001	Extremo de métricas de Prometheus
8000	Servidor REST de Trident
17546	Puerto de sonda de presencia/preparación utilizado por los pods demonset de Trident

El puerto de la sonda de nivel de gravedad/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 DE REST de Astra Trident

Aunque "comandos y opciones de trimentctl" Son la forma más sencilla de interactuar con la API REST de Astra Trident, puede utilizar el extremo REST directamente si lo prefiere.

Cuándo utilizar la API DE REST

La API REST es útil en las instalaciones avanzadas que usan Astra Trident como binario independiente en las puestas en marcha sin Kubernetes.

Para una mayor seguridad, la Astra Trident `REST API` se restringe a localhost de forma predeterminada cuando se ejecuta dentro de un pod. Para cambiar este comportamiento, debe configurar Astra Trident's. `-address` en su configuración del pod.

Uso de la API DE REST

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 con nombre.

`POST`

`POST <trident-address>/trident/v1/<object-type>`: Crea un objeto del tipo especificado.

- Requiere la configuración de JSON para el objeto que se cree. Para obtener información sobre la especificación de cada tipo de objeto, consulte LINK:tridentctl.html[`tridentctl` comandos y opciones].
- Si el objeto ya existe, el comportamiento varía: Los back-ends actualizan el objeto existente, mientras que todos los demás tipos de objeto fallarán la operación.

`DELETE`

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`: Elimina el recurso con nombre.

> Seguirán existiendo volúmenes asociados con back-ends o clases de almacenamiento, que deben eliminarse por separado. Para obtener más información, consulte el enlace:tridentctl.html[`tridentctl` comandos y opciones].

Para obtener ejemplos de cómo se llama a estas API, pase la depuración (`-d`) bandera. Para obtener más información, consulte el enlace:tridentctl.html[`tridentctl` comandos y opciones].

Opciones de línea de comandos

Astra Trident expone varias opciones de línea de comandos para Trident orchestrator. Puede usar estas opciones para modificar la 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`: Utilice esta opción o. `-k8s_api_server` Para habilitar la compatibilidad con Kubernetes. Al configurar esto, Trident usa las credenciales de cuenta del servicio de Kubernetes del pod para contactar con 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>`: Utilice esta opción o. `-k8s_pod` Para habilitar la compatibilidad con Kubernetes. Cuando se especifica, Trident se conecta al servidor API de Kubernetes mediante el puerto y la dirección no seguras que se proporcionan. Esto permite que Trident se ponga en marcha fuera de un pod; sin embargo, solo admite conexiones no seguras con el servidor API. Para conectarse con seguridad, implemente Trident en un pod con el `-k8s_pod` opción.
- `-k8s_config_path <file>`: Necesario; debe especificar esta ruta de acceso a un archivo KubeConfig.

Docker

- `-volume_driver <name>`: Nombre del controlador utilizado al registrar el complemento Docker. De forma predeterminada es `netapp`.

- `-driver_port <port-number>`: Escucha en este puerto en lugar de un socket de dominio UNIX.
- `-config <file>`: Necesario; debe especificar esta ruta de acceso a un archivo de configuración de back-end.

DESCANSO

- `-address <ip-or-host>`: Especifica la dirección en la que debe escuchar el servidor REST de Trident. El valor predeterminado es localhost. Cuando se escucha en localhost y se ejecuta dentro de un pod Kubernetes, la interfaz REST no es accesible desde fuera del pod. Uso `-address ""` Para hacer que la interfaz DE REST sea accesible desde la dirección IP del pod.

> La interfaz DE REST de Trident se puede configurar para escuchar y servir únicamente 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. El valor predeterminado es 8000.
- `-rest`: Activa la interfaz DE REPOSO. El valor predeterminado es TRUE.

Los productos de NetApp están integrados con Kubernetes

La cartera de productos de almacenamiento de NetApp se integra con muchos aspectos diferentes de un clúster de Kubernetes, por lo que proporciona funcionalidades de gestión de datos avanzadas que mejoran la funcionalidad, la funcionalidad, el rendimiento y la disponibilidad de la puesta en marcha de Kubernetes.

Astra

"Astra" Facilita a las empresas la gestión, protección y movimiento de sus cargas de trabajo en contenedores con gran cantidad de datos que se ejecutan en Kubernetes en los clouds públicos y en las instalaciones. Astra aprovisiona y proporciona almacenamiento en contenedores persistente mediante Trident de la cartera de almacenamiento NetApp demostrada y expansiva en el cloud público y en las instalaciones. También ofrece un conjunto amplio de funcionalidades avanzadas de gestión de datos para aplicaciones, como snapshots, backups y restauración, registros de actividades y clonado activo para la protección de datos, recuperación ante desastres/datos, auditoría de datos y casos de uso de migración para cargas de trabajo de Kubernetes.

ONTAP

ONTAP es el sistema operativo de almacenamiento unificado y multiprotocolo de NetApp que ofrece funcionalidades avanzadas de gestión de datos para cualquier aplicación. Los sistemas ONTAP tienen configuraciones all-flash, híbridas o all-HDD y ofrecen muchos modelos de puesta en marcha diferentes, como hardware a medida (FAS y AFF), unidad genérica (ONTAP Select) y solo cloud (Cloud Volumes ONTAP).

> Trident es compatible con todos los modelos de puesta en marcha de ONTAP mencionados anteriormente.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" Es un dispositivo de almacenamiento exclusivamente de software que ejecuta el software para la gestión de datos ONTAP en el cloud. Puede utilizar Cloud Volumes ONTAP para cargas de

trabajo de producción, recuperación ante desastres, DevOps, recursos compartidos de archivos y gestión de bases de datos. Amplía el almacenamiento empresarial al cloud ofreciendo eficiencias del almacenamiento, alta disponibilidad, replicación de datos, organización en niveles de los datos y consistencia de las aplicaciones.

Amazon FSX para ONTAP de NetApp

"Amazon FSX para ONTAP de NetApp" Es un servicio AWS totalmente gestionado que le permite iniciar y ejecutar sistemas de archivos con tecnología del sistema operativo de almacenamiento NetApp ONTAP. FSX para ONTAP te permite aprovechar las funciones, el rendimiento y las funcionalidades administrativas de NetApp que ya conoces, a la vez que aprovechas la simplicidad, la agilidad, la seguridad y la escalabilidad de almacenar datos en AWS. FSX para ONTAP es compatible con muchas de las funciones del sistema de archivos ONTAP y las API de administración.

Software Element

"Elemento" permite al administrador de almacenamiento consolidar cargas de trabajo garantizando el rendimiento y haciendo posible un espacio de almacenamiento simplificado y optimizado. Junto con una API para permitir la automatización de todos los aspectos de la gestión del almacenamiento, Element permite a los administradores de almacenamiento hacer más con menos esfuerzo.

NetApp HCI

"NetApp HCI" simplifica la gestión y el escalado del centro de datos mediante la automatización de las tareas rutinarias y permite que los administradores de la infraestructura se centren en funciones más importantes.

Trident es totalmente compatible con NetApp HCI. Trident puede aprovisionar y gestionar dispositivos de almacenamiento para aplicaciones en contenedores directamente en la plataforma de almacenamiento subyacente de NetApp HCI.

Azure NetApp Files

"Azure NetApp Files" Es un servicio de recursos compartidos de archivos de Azure de clase empresarial con la tecnología de NetApp. Puede ejecutar sus cargas de trabajo basadas en archivos más exigentes de forma nativa en Azure, con el rendimiento y la gestión de datos enriquecidos que espera de NetApp.

Cloud Volumes Service para Google Cloud

"Cloud Volumes Service de NetApp para Google Cloud" Es un servicio de archivos nativo del cloud que proporciona volúmenes de NAS en NFS y SMB con rendimiento all-flash. Este servicio permite que se ejecute cualquier carga de trabajo, incluidas las aplicaciones heredadas, en la nube de GCP. Proporciona un servicio totalmente gestionado que ofrece alto rendimiento consistente, clonado instantáneo, protección de datos y acceso seguro a instancias de Google Compute Engine (GCE).

Objetos de Kubernetes y Trident

Puede interactuar con Kubernetes y Trident mediante las API DE REST a través de la lectura y la escritura de 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 mediante Kubernetes y los demás se gestionan mediante Trident.

¿Cómo interactúan los objetos entre sí?

Quizás la forma más sencilla de comprender los objetos, qué hacen y cómo interactúan sea, es seguir una única solicitud de almacenamiento a un usuario de Kubernetes:

1. Un usuario crea un `PersistentVolumeClaim` solicitando un nuevo `PersistentVolume` De un tamaño concreto de un Kubernetes `StorageClass` previamente configurado por el administrador.
2. Kubernetes `StorageClass` Identifica a Trident como su aprovisionador y incluye los parámetros que indican a Trident cómo aprovisionar un volumen para la clase solicitada.
3. Trident analiza sus propios recursos `StorageClass` con el mismo nombre que identifica la coincidencia `Backends` y.. `StoragePools` que puede usar para aprovisionar volúmenes para la clase.
4. Trident aprovisiona el almacenamiento en un back-end coincidente y crea dos objetos: Un `PersistentVolume` En Kubernetes, donde se indica cómo encontrar, montar y tratar el volumen, y un volumen en Trident que conserva la relación entre `PersistentVolume` y el almacenamiento real.
5. Kubernetes enlaza con el `PersistentVolumeClaim` a los nuevos `PersistentVolume`. Pods que incluyen `PersistentVolumeClaim` monte ese volumen persistente en cualquier host en el que se ejecute.
6. Un usuario crea un `VolumeSnapshot` De un PVC existente, utilizando un `VolumeSnapshotClass` Eso es lo que apunta a Trident.
7. Trident identifica el volumen asociado con la RVP y crea una copia Snapshot del volumen en su back-end. También crea un `VolumeSnapshotContent` Esto indica a Kubernetes cómo identificar la snapshot.
8. Un usuario puede crear un `PersistentVolumeClaim` uso `VolumeSnapshot` como origen.
9. Trident identifica la instantánea necesaria y realiza el mismo conjunto de pasos involucrados en la creación de un `PersistentVolume` y un `Volume`.

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

Kubernetes `PersistentVolumeClaim` objetos

Un Kubernetes `PersistentVolumeClaim` El objeto es una solicitud de almacenamiento que realiza un usuario de 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 de back-end:

Anotación	Opción de volumen	Controladores compatibles
trident.netapp.io/fileSystem	Sistema de archivos	ontap-san, solidfire-san, ontap-san-economy
trident.netapp.io/cloneFromPVC	ClonSourceVolume	ontap-nas ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economía
trident.netapp.io/splitOnClone	SplitOnClone	ontap-nas y ontap-san
trident.netapp.io/protocol	protocolo	cualquiera

Anotación	Opción de volumen	Controladores compatibles
trident.netapp.io/exportPolicy	Política de exportoPolicy	ontap-nas ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Política de copias Snapshot	ontap-nas ontap-nas-economy, ontap-nas-flexgroup y ontap-san
trident.netapp.io/snapshotReserve	Reserva de copias Snapshot	ontap-nas ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/snapshotDirectory	Snapshot shotDirectory	ontap-nas ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	Permisos univalados	ontap-nas ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	Tamaño del bloque	solidfire-san

Si el VP creado tiene el `Delete` Reclamar política, Trident elimina el VP y el volumen de respaldo cuando se libera el VP (es decir, cuando el usuario elimina la RVP). Si la acción de eliminación falla, Trident Marca el VP como tal y reintenta periódicamente la operación hasta que esta se complete o se elimine manualmente el VP. Si el VP utiliza `Retain` Política, Trident ignora la operación y asume que el administrador la limpiará desde Kubernetes y el back-end, lo que permitirá realizar un backup o la inspección del volumen antes de su eliminación. Tenga en cuenta que al eliminar el VP, Trident no eliminará el volumen de backup. Debe quitarlo usando la API DE REST (`tridentctl`).

Trident admite la creación de instantáneas de volumen utilizando la especificación CSI: Puede crear una instantánea de volumen y utilizarla como origen de datos para clonar las RVP existentes. De este modo, las copias puntuales de VP pueden exponerse a Kubernetes en forma de snapshots. Las instantáneas pueden utilizarse para crear nuevos VP. Eche un vistazo `On-Demand Volume Snapshots` para ver cómo funcionaría.

Trident también proporciona la `cloneFromPVC` y.. `splitOnClone` anotaciones para crear clones. Puede utilizar estas anotaciones para clonar una RVP sin tener que utilizar la implementación de CSI.

A continuación se muestra un ejemplo: Si un usuario ya tiene una RVP llamada `mysql`, El usuario puede crear un nuevo PVC llamado `mysqlclone` mediante la anotación, por ejemplo `trident.netapp.io/cloneFromPVC: mysql`. Con este conjunto de anotaciones, Trident clona el volumen correspondiente a la RVP de mysql, en lugar de aprovisionar un volumen desde cero.

Considere los siguientes puntos:

- Se recomienda clonar un volumen inactivo.
- Una RVP y su clon deben estar en el mismo espacio de nombres de Kubernetes y tener el mismo tipo de almacenamiento.
- Con la `ontap-nas` y.. `ontap-san` Controladores, es posible que sea conveniente establecer la anotación de PVC `trident.netapp.io/splitOnClone` en conjunto con `trident.netapp.io/cloneFromPVC`. Con `trident.netapp.io/splitOnClone` establezca en

`true`, Trident divide el volumen clonado del volumen principal y, por lo tanto, separa completamente el ciclo de vida del volumen clonado de su principal a costa de perder alguna eficiencia de almacenamiento. No está configurado `trident.netapp.io/splitOnClone` o establecerlo en `false` provoca una reducción del consumo de espacio en el back-end a costa de crear dependencias entre los volúmenes principal y clonado, de modo que no se pueda eliminar el volumen principal, a menos que el clon se elimine primero. Una situación en la que dividir el clon tiene sentido es clonar un volumen de base de datos vacío donde se espera que tanto el volumen como su clon desvíen enormemente y no se beneficien de las eficiencias del almacenamiento ofrecidas por ONTAP.

La `sample-input` el directorio contiene ejemplos de definiciones de PVC para utilizarlas con Trident. Consulte Para obtener una descripción completa de los parámetros y la configuración asociados con los volúmenes de Trident.

Kubernetes `PersistentVolume` objetos

Un Kubernetes `PersistentVolume` Object representa un fragmento de almacenamiento que se pone a disposición del clúster de Kubernetes. Tiene un ciclo de vida independiente del pod que lo utiliza.

Crea Trident `PersistentVolume` Los objetos y los registra automáticamente con el clúster Kubernetes en función de los volúmenes que aprovisiona. No se espera que usted los gestione usted mismo.

Cuando se crea una RVP que hace referencia a un sistema basado en Trident `StorageClass`, Trident aprovisiona un nuevo volumen utilizando la clase de almacenamiento correspondiente y registra un nuevo VP para ese volumen. Al configurar el volumen aprovisionado y el VP correspondiente, Trident sigue las siguientes reglas:

- Trident genera un nombre PV para Kubernetes y un nombre interno que utiliza para aprovisionar el almacenamiento. En ambos casos, se asegura de que los nombres son únicos en su alcance.
- El tamaño del volumen coincide con el tamaño solicitado en el PVC lo más cerca posible, aunque podría redondearse a la cantidad más cercana asignable, dependiendo de la plataforma.

Kubernetes `StorageClass` objetos

Kubernetes `StorageClass` los objetos se especifican por nombre en `PersistentVolumeClaims` para aprovisionar el almacenamiento con una serie de propiedades. La clase de almacenamiento identifica el aprovisionador que se usará y define ese conjunto de propiedades en términos que entiende el aprovisionador.

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

Un Kubernetes `StorageClass` Objeto que usa Trident tiene el siguiente 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 y dicen a Trident cómo aprovisionar volúmenes para la clase.

Los parámetros de la clase de almacenamiento son:

Atributo	Tipo	Obligatorio	Descripción
atributos	map[string]string	no	Consulte la sección atributos a continuación
Pools de almacenamiento	Map[string]StringList	no	Mapa de nombres de backend a listas de pools de almacenamiento dentro
AdicionalStoragePools	Map[string]StringList	no	Mapa de nombres de backend a listas de pools de almacenamiento de
ExcludeStoragePools	Map[string]StringList	no	Asignación de nombres de backend a. listas de pools de almacenamiento en

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

Atributos de selección del pool de almacenamiento

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

Atributo	Tipo	Valores	Oferta	Solicitud	Admitido por
media 1	cadena	hdd, híbrido, ssd	Pool contiene medios de este tipo; híbrido significa ambos	Tipo de medios especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san y solidfire-san

Atributo	Tipo	Valores	Oferta	Solicitud	Admitido por
AprovisionaciónTipo	cadena	delgado, grueso	El pool admite este método de aprovisionamiento	Método de aprovisionamiento especificado	grueso: all ONTAP; thin: all ONTAP y solidfire-san
Tipo de backendType	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
snapshot	bool	verdadero, falso	El pool admite volúmenes con Snapshot	Volumen con snapshots habilitadas	ontap-nas, ontap-san, solidfire-san y gcp-cvs
clones	bool	verdadero, falso	Pool admite el clonado de volúmenes	Volumen con clones habilitados	ontap-nas, ontap-san, solidfire-san y gcp-cvs
cifrado	bool	verdadero, falso	El pool admite volúmenes cifrados	Volumen con cifrado habilitado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	entero positivo	El pool es capaz de garantizar IOPS en este rango	El volumen garantizado de estas IOPS	solidfire-san

Esta versión 1: No es compatible con sistemas ONTAP Select

En la mayoría de los casos, los valores solicitados influyen directamente en el aprovisionamiento; por ejemplo, solicitar un aprovisionamiento de alto rendimiento da lugar a un volumen considerablemente aprovisionado. Sin embargo, un pool de almacenamiento de Element utiliza el valor mínimo y máximo de IOPS que ofrece para establecer los valores de calidad de servicio, en lugar del valor solicitado. En este caso, el valor solicitado se utiliza solo para seleccionar el pool de almacenamiento.

Lo ideal es que pueda usar `attributes` solo para modelar las cualidades del almacenamiento que necesita para satisfacer las necesidades de una clase particular. Trident detecta y selecciona automáticamente pools de almacenamiento que coincidan *all* del `attributes` que especifique.

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

Puede utilizar el `storagePools` el parámetro para restringir aún más el conjunto de pools que coinciden con cualquier especificado `attributes`. En otras palabras, Trident utiliza la intersección de pools identificados por el `attributes` y.. `storagePools` parámetros para el aprovisionamiento. Es posible usar un parámetro solo o ambos juntos.

Puede utilizar el `additionalStoragePools` Parámetro para ampliar el conjunto de pools que Trident utiliza para el aprovisionamiento, independientemente de cualquier pool que seleccione `attributes` y.. `storagePools` parámetros.

Puede utilizar el `excludeStoragePools` Parámetro para filtrar el conjunto de pools que Trident utiliza para el aprovisionamiento. Cuando se usa este parámetro, se quitan todos los pools que coinciden.

En la `storagePools` y.. `additionalStoragePools` parámetros, cada entrada toma el formulario `<backend>:<storagePoolList>`, donde `<storagePoolList>` es una lista de pools de almacenamiento separados por comas para el back-end especificado. Por ejemplo, un valor para `additionalStoragePools` puede parecer `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Estas listas aceptan valores regex para los valores de backend y list. Puede utilizar `tridentctl get backend` para obtener la lista de los back-ends y sus pools.

Atributos de Kubernetes

Trident no afecta a la selección de pools y back-ends de almacenamiento durante el aprovisionamiento dinámico. En su lugar, estos atributos simplemente ofrecen 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	Controladores relevantes	Kubernetes Versión
Tipo fstype	cadena	ext4, ext3, xfs, etc.	Tipo de sistema de archivos para el bloque volúmenes	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economía	Todo
Expansión de allowVolume	booleano	verdadero, falso	Habilite o deshabilite el soporte para aumentar el tamaño de PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1,11 o posterior

VolumeBindingMode	cadena	Inmediatamente, WaitForFirstConsumer	Elija cuándo se producen el enlace de volumen y el aprovisionamiento dinámico	Todo	1,19 - 1,26

- La `fsType` El parámetro se utiliza para controlar el tipo de sistema de archivos deseado para las LUN DE SAN. Además, Kubernetes utiliza también 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 la `fsGroup` contexto de seguridad de un pod solo if `fsType` está configurado. Consulte "Kubernetes: Configure un contexto de seguridad para un Pod o contenedor" para obtener información general sobre la configuración de la propiedad del volumen con `fsGroup` contexto. Kubernetes aplicará el `fsGroup` valor 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 utilizar `fsGroup` la clase de almacenamiento aún debe especificar un `fsType`. Puede configurarlo en `nfs` o cualquier valor que no sea nulo.

- Consulte "Expanda los volúmenes" para obtener más información sobre la expansión de volumen.
- El paquete de instalación de Trident proporciona varias definiciones de clase de almacenamiento de ejemplo para usar con Trident en `sample-input/storage-class-*.yaml`. Al eliminar una clase de almacenamiento Kubernetes, también se elimina el tipo de almacenamiento Trident correspondiente.

Kubernetes `VolumeSnapshotClass` objetos

Kubernetes `VolumeSnapshotClass` los objetos son similares `StorageClasses`. Ayudan a definir varias clases de almacenamiento y las instantáneas de volumen hacen referencia a ellas para asociar la snapshot a la clase de snapshot necesaria. Cada copia de Snapshot de volumen se asocia con una sola clase de copia de Snapshot de volumen.

1. `VolumeSnapshotClass` debe ser definido por un administrador para crear snapshots. Una clase de snapshot 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
```

La `driver` Especifica a Kubernetes que solicitudes de snapshots de volumen del `csi-snapclass` Trident gestiona la clase. La `deletionPolicy` especifica la acción que se debe realizar cuando se debe eliminar

una instantánea. Cuando `deletionPolicy` se establece en `Delete`, los objetos de instantánea del volumen, así como la instantánea subyacente en el clúster de almacenamiento, se eliminan cuando se elimina una instantánea. Como alternativa, establecerlo en `Retain` significa eso `VolumeSnapshotContent` y se conserva la snapshot física.

Kubernetes `VolumeSnapshot` objetos

Un Kubernetes `VolumeSnapshot` objeto es una solicitud para crear una copia de snapshot de un volumen. Del mismo modo que la RVP representa una solicitud al usuario para un volumen, un snapshot de volumen es una solicitud al que hace un usuario para crear una copia Snapshot de una RVP existente.

Cuando llega una solicitud Snapshot de volumen, Trident gestiona automáticamente la creación de la snapshot para el volumen en el back-end y expone la snapshot creando un único `VolumeSnapshotContent` objeto. Puede crear instantáneas a partir de EVs existentes y utilizar las instantáneas como DataSource al crear nuevas CVP.

> El ciclo de vida de un VolumeSnapshot es independiente del PVC de origen: Una instantánea persiste incluso después de eliminar el PVC de origen. Cuando se elimina un PVC que tiene instantáneas asociadas, Trident Marca el volumen de respaldo de este PVC con el estado **Eliminación**, pero no lo elimina por completo. El volumen se elimina cuando se eliminan todas las Snapshot asociadas.

Kubernetes `VolumeSnapshotContent` objetos

Un Kubernetes `VolumeSnapshotContent` object representa una snapshot tomada de un volumen ya aprovisionado. Es similar a un `PersistentVolume` y significa una instantánea aprovisionada en el clúster de almacenamiento. Similar a. `PersistentVolumeClaim` y.. `PersistentVolume` los objetos, cuando se crea una snapshot, el `VolumeSnapshotContent` object mantiene una asignación de uno a uno `VolumeSnapshot` objeto, que solicitó la creación de la snapshot.

La `VolumeSnapshotContent` el objeto contiene detalles que identifican de manera única la instantánea, como la `snapshotHandle`. Este `snapshotHandle` Es una combinación única del nombre del PV y el nombre del `VolumeSnapshotContent` objeto.

Cuando llega una solicitud de Snapshot, Trident crea la snapshot en el back-end. Una vez creada la copia de Snapshot, Trident configura un `VolumeSnapshotContent` Objeto y, por lo tanto, expone la snapshot a la API de Kubernetes.

> Por lo general, no es necesario gestionar el `VolumeSnapshotContent` objeto. Una excepción a esto es cuando lo desea "importe una copia de snapshot de volumen" Creado fuera de Astra Trident.

Kubernetes `CustomResourceDefinition` objetos

Los recursos personalizados de Kubernetes son extremos 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 un conjunto de objetos. Puede obtener estas definiciones de recursos ejecutando `kubectl get crds`.

Kubernetes almacena en su almacén de metadatos las definiciones de recursos personalizadas (CRD) y los metadatos de objetos asociados. De este modo, no es necesario disponer de un almacén aparte para Trident.

Usos de Astra Trident `CustomResourceDefinition` Objetos que conservan la identidad de objetos de Trident, como los back-ends de Trident, las clases de almacenamiento de Trident y los volúmenes de Trident. Trident gestiona estos objetos. Además, el marco de instantáneas de volumen CSI introduce algunos CRD necesarios para definir instantáneas de volumen.

Los multos son una estructura de Kubernetes. Trident crea los objetos de los recursos definidos anteriormente. Como ejemplo simple, cuando se crea un back-end usando `tridentctl`, a correspondiente `tridentbackends` El objeto CRD se crea para el consumo por parte de Kubernetes.

A continuación se indican algunos puntos que hay que tener en cuenta sobre los CRD de Trident:

- Cuando se instala Trident, se crea un conjunto de CRD que se puede utilizar como cualquier otro tipo de recurso.
- Al desinstalar Trident mediante la `tridentctl uninstall` Comando, los pods de Trident se eliminan, pero los CRD creados no se borran. Consulte "Desinstale Trident" Para comprender cómo Trident se puede eliminar por completo y volver a configurar desde cero.

Astra Trident `StorageClass` objetos

Trident crea clases de almacenamiento coincidentes para Kubernetes `StorageClass` objetos que especifican `csi.trident.netapp.io/netapp.io/trident` 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 crea un Kubernetes `StorageClass` Que usa Trident como aprovisionador está registrado.

Las clases de almacenamiento comprenden un conjunto de requisitos para los volúmenes. Trident enlaza estos requisitos con los atributos presentes en cada pool de almacenamiento; si coinciden, ese pool de almacenamiento es un objetivo válido para aprovisionar volúmenes que utilizan esa clase de almacenamiento.

Puede crear configuraciones de clase de almacenamiento para definir clases de almacenamiento directamente mediante la API DE REST. Sin embargo, en el caso de las puestas en marcha de Kubernetes, esperamos que se creen al registrar el nuevo Kubernetes `StorageClass` objetos.

Objetos back-end de Astra Trident

Los back-ends representan a los proveedores de almacenamiento, además de los cuales Trident aprovisiona volúmenes; una única instancia de Trident puede gestionar cualquier número de back-ends.

Éste es uno de los dos tipos de objeto que se crean y administran a sí mismo. El otro es Kubernetes `StorageClass` objeto.

Para obtener más información acerca de cómo construir estos objetos, consulte "configuración de los back-ends".

Astra Trident `StoragePool` objetos

Los pools de almacenamiento representan las distintas ubicaciones disponibles para aprovisionar en cada back-end. Para ONTAP, corresponden a los agregados en las SVM. Para HCI/SolidFire de NetApp, corresponden a las bandas de calidad de servicio especificadas por el administrador. Para Cloud Volumes Service, se corresponden con las regiones de proveedores de cloud. Cada pool de almacenamiento tiene un

conjunto de atributos de almacenamiento distintos que definen sus características de rendimiento y sus características de protección de datos.

Al contrario de lo que ocurre con otros objetos aquí, los candidatos de pools de almacenamiento siempre se detectan y gestionan automáticamente.

Astra Trident `Volume` objetos

Los volúmenes son la unidad básica de aprovisionamiento y constan de extremos back-end, como recursos compartidos de NFS y LUN iSCSI. En Kubernetes, se corresponden directamente con `PersistentVolumes`. Cuando crea un volumen, asegúrese de que tiene una clase de almacenamiento, que determina dónde se puede aprovisionar ese volumen junto con un tamaño.

- En Kubernetes, estos objetos se gestionan automáticamente. Es posible verlos para ver qué ha aprovisionado Trident.
- Al eliminar un VP con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un estado **Eliminación**. Para que se elimine el volumen de Trident, es necesario quitar las snapshots del volumen.

Una configuración de volumen define las propiedades que debe tener un volumen aprovisionado.

Atributo	Tipo	Obligatorio	Descripción
versión	cadena	no	Versión de la API de Trident ("1")
nombre	cadena	sí	Nombre del volumen que se va a crear
Clase de almacenamiento	cadena	sí	Clase de almacenamiento que se utilizará al aprovisionar el volumen
tamaño	cadena	sí	El tamaño del volumen que se va a aprovisionar en bytes
protocolo	cadena	no	Tipo de protocolo que se va a utilizar; "archivo" o "bloque"
InternalName	cadena	no	Nombre del objeto en el sistema de almacenamiento, generado por Trident
ClonSourceVolume	cadena	no	ONTAP (nas, san) y SolidFire-*: Nombre del volumen desde el que se va a clonar
SplitOnClone	cadena	no	ONTAP (nas, san): Divida el clon entre su primario
Política de copias Snapshot	cadena	no	ONTAP-*: Política de instantánea a utilizar

Atributo	Tipo	Obligatorio	Descripción
Reserva de copias Snapshot	cadena	no	ONTAP-*: Porcentaje del volumen reservado para instantáneas
Política de exportoPolicy	cadena	no	ontap-nas*: Política de exportación que se va a utilizar
Snapshot shotDirectory	bool	no	ontap-nas*: Si el directorio de instantáneas está visible
Permisos univalados	cadena	no	ontap-nas*: Permisos iniciales de UNIX
Tamaño del bloque	cadena	no	SolidFire-*: Tamaño de bloque/sector
Sistema de archivos	cadena	no	Tipo de sistema de archivos

Genera Trident `internalName` al crear el volumen. Esto consta de dos pasos. En primer lugar, prepens el prefijo de almacenamiento (ya sea el predeterminado) `trident` o el prefijo de la configuración del back-end) al nombre del volumen, lo que genera el nombre del formulario `<prefix>-<volume-name>`. A continuación, procede a desinfectar el nombre y a reemplazar los caracteres no permitidos en el backend. En los back-ends de ONTAP, reemplaza guiones con guiones bajos (de esta forma, el nombre interno se convierte en `<prefix>_<volume-name>`). En los back-ends de Element, reemplaza guiones bajos por guiones.

Puede utilizar configuraciones de volumen para aprovisionar directamente los volúmenes mediante la API REST, pero en las puestas en marcha de Kubernetes esperamos que la mayoría de los usuarios usen el Kubernetes estándar `PersistentVolumeClaim` método. Trident crea este objeto de volumen de forma automática como parte del aprovisionamiento
proceso.

Astra Trident `Snapshot` objetos

Las Snapshot son una copia de un momento específico de los volúmenes, que se pueden usar para aprovisionar nuevos volúmenes o restaurar el estado. En Kubernetes, se corresponden directamente con `VolumeSnapshotContent` objetos. Cada copia de Snapshot se asocia con un volumen, que es el origen de los datos de la copia de Snapshot.

Cada uno `Snapshot` object incluye las propiedades que se enumeran a continuación:

Atributo	Tipo	Obligatorio	Descripción
versión	Cadena	Sí	Versión de la API de Trident ("1")
nombre	Cadena	Sí	Nombre del objeto Snapshot de Trident

Atributo	Tipo	Obligatorio	Descripción
InternalName	Cadena	Sí	Nombre del objeto Snapshot de Trident en el sistema de almacenamiento
Nombre de volumen	Cadena	Sí	Nombre del volumen persistente para el que se crea la snapshot
VolumeInternalName	Cadena	Sí	Nombre del objeto de volumen de Trident asociado en el sistema de almacenamiento

En Kubernetes, estos objetos se gestionan automáticamente. Es posible verlos para ver qué ha aprovisionado Trident.

Cuando un Kubernetes `VolumeSnapshot` Se crea la solicitud del objeto, Trident funciona mediante la creación de un objeto Snapshot en el sistema de almacenamiento que realiza backups. La `internalName` de este objeto snapshot se genera combinando el prefijo `snapshot-` con la `UID` de la `VolumeSnapshot` objeto (por ejemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` y.. `volumeInternalName` se rellenan obteniendo los detalles del respaldo volumen.

Astra Trident `ResourceQuota` objeto

El inicio de Trident consume un `system-node-critical` Clase de prioridad, la clase de prioridad más alta disponible en Kubernetes, para garantizar que Astra Trident pueda identificar y limpiar volúmenes durante un apagado correcto de nodos y permitir que Trident demonset pods prevea las cargas de trabajo con una prioridad menor en clústeres donde hay una alta presión en los recursos.

Para conseguirlo, Astra Trident utiliza una `ResourceQuota` Objeto garantizar que se cumple una clase prioritaria "system-node-Critical" en el demonset de Trident. Antes de la puesta en marcha y la creación de demonset, Astra Trident busca la `ResourceQuota` object y, si no se detecta, lo aplica.

Si necesita más control sobre la cuota de recursos predeterminada y la clase de prioridad, puede generar una `custom.yaml` o configure el `ResourceQuota` Objeto mediante el gráfico Helm.

A continuación se muestra un ejemplo de un objeto "ResourceQuota’object que da prioridad al demonset de 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 acerca de las cuotas de recursos, consulte "Kubernetes: Cuotas de recursos".

Limpie `ResourceQuota` si la instalación falla

En el raro caso en que la instalación falle después del `ResourceQuota` se crea el objeto, primero se intenta "desinstalando" y, a continuación, vuelva a instalar.

Si esto no funciona, quite manualmente la `ResourceQuota` objeto.

Quitar `ResourceQuota`

Si prefiere controlar su propia asignación de recursos, puede eliminar Astra Trident `ResourceQuota` objeto con el comando:

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

comandos y opciones de trimentctl

La "Paquete de instalación de Trident" incluye una utilidad de línea de comandos, `tridentctl`, Que proporciona un acceso sencillo a Astra Trident. Los usuarios de Kubernetes con suficientes privilegios pueden usarlo para instalar Astra Trident y también para interactuar con ella directamente para gestionar el espacio de nombres que contiene el pod Astra Trident.

Comandos y opciones disponibles

Para obtener información de uso, ejecute `tridentctl --help`.

Los comandos disponibles y las opciones globales son:

```
Usage:
  tridentctl [command]
```

Comandos disponibles:

- `create`: Añadir un recurso a Astra Trident.
- `delete`: Elimine uno o más recursos de Astra Trident.
- `get`: Obtenga uno o más recursos de Astra Trident.
- `help`: Ayuda sobre cualquier comando.
- `images`: Imprime una tabla de las imágenes de contenedores que necesita Astra Trident.
- `import`: Importe un recurso existente a Astra Trident.
- `install`: Instalar Astra Trident.
- `logs`: Imprime los registros de Astra Trident.
- `send`: Envíe un recurso desde Astra Trident.
- `uninstall`: Desinstalar Astra Trident.
- `update`: Modificar un recurso en Astra Trident.
- `upgrade`: Actualizar un recurso en Astra Trident.
- `version`: Imprime la versión de Astra Trident.

Indicadores:

- `-d, --debug`: Salida de depuración.
- `-h, --help`: Ayuda para `tridentctl`.
- `-n, --namespace string`: Espacio de nombres de la implementación de Astra Trident.
- `-o, --output string`: Formato de salida. Uno de json|yaml|name|Wide|ps (predeterminado).
- `-s, --server string`: Dirección/puerto de la interfaz REST de Astra Trident.

> La interfaz DE REST de Trident se puede configurar para escuchar y servir únicamente en 127.0.0.1 (para IPv4) o [::1] (para IPv6).

> La interfaz DE REST de Trident se puede configurar para escuchar y servir únicamente en 127.0.0.1 (para IPv4) o [::1] (para IPv6).

`create`

Puede utilizar ejecutar el `create` Comando para añadir un recurso a Astra Trident.

```
Usage:
  tridentctl create [option]
```

Opción disponible:
`backend`: Añadir un back-end a Astra Trident.

`delete`

Puede ejecutar el `delete` Comando para eliminar uno o más recursos de Astra Trident.

```
Usage:
  tridentctl delete [option]
```

Opciones disponibles:

- `backend`: Elimine uno o más back-ends de almacenamiento de Astra Trident.
- `snapshot`: Elimine una o más instantáneas de volumen de Astra Trident.
- `storageclass`: Elimine una o varias clases de almacenamiento de Astra Trident.
- `volume`: Elimine uno o varios volúmenes de almacenamiento de Astra Trident.

`get`

Puede ejecutar el `get` Comando para obtener uno o más recursos de Astra Trident.

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

Opciones disponibles:

- `backend`: Obtenga uno o más back-ends de almacenamiento de Astra Trident.
- `snapshot`: Obtiene una o más instantáneas de Astra Trident.
- `storageclass`: Obtenga una o más clases de almacenamiento de Astra Trident.
- `volume`: Obtenga uno o más volúmenes de Astra Trident.

`volume` indicadores:
* `-h, --help`: Ayuda para volúmenes.
* `--parentOfSubordinate string`: Limite la consulta al volumen de origen subordinado.
* `--subordinateOf string`: Limite la consulta a las subordinadas del volumen.

`images`

Puede ejecutar el `images` Indicador para imprimir una tabla de las imágenes de contenedor que necesita

Astra Trident.

```
Usage:
  tridentctl images [flags]
```

Indicadores:
* `-h, --help`: Help for images.
* -V, --k8s-version string': Versión semántica del cluster de Kubernetes.

import volume

Puede ejecutar el `import volume` Comando para importar un volumen existente a Astra Trident.

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

Alias:
`volume, v`

Indicadores:

- `-f, --filename string`: Ruta al archivo YLMA o JSON PVC.
- `-h, --help`: Ayuda para el volumen.
- `--no-manage`: Cree sólo PV/PVC. No asuma que se gestiona el ciclo de vida de los volúmenes.

install

Puede ejecutar el `install` Banderas para instalar Astra Trident.

```
Usage:
  tridentctl install [flags]
```

Indicadores:

- `--autosupport-image string`: La imagen de contenedor para la telemetría de AutoSupport (valor predeterminado «netapp/trident autosupport:<current-version>»).
- `--autosupport-proxy string`: La dirección/puerto de un proxy para enviar telemetría AutoSupport.
- `--enable-node-prep`: Intente instalar los paquetes necesarios en los nodos.
- `--generate-custom-yaml`: Genere archivos YAML sin instalar nada.
- `-h, --help`: Ayuda para instalar.
- `--http-request-timeout`: Sustituya el tiempo de espera de la solicitud HTTP para la API REST del controlador Trident (por defecto 1m30s).

- `--image-registry string`: La dirección/puerto de un registro de imagen interna.
- `--k8s-timeout duration`: El tiempo de espera para todas las operaciones de Kubernetes (por defecto 3 m0s).
- `--kubelet-dir string`: La ubicación del host del estado interno de Kubelet (predeterminado "/var/lib/kubelet").
- `--log-format string`: El formato de registro de Astra Trident (texto, json) (por defecto "text").
- `--pv string`: El nombre del PV heredado utilizado por Astra Trident, se asegura de que esto no existe (por defecto "trident").
- `--pvc string`: El nombre del PVC heredado utilizado por Astra Trident, se asegura de que esto no exista (por defecto “tridente”).
- `--silence-autosupport`: No envíe los paquetes AutoSupport a NetApp automáticamente (valor predeterminado: TRUE).
- `--silent`: Desactiva la mayoría de la salida durante la instalación.
- `--trident-image string`: La imagen de Astra Trident que se va a instalar.
- `--use-custom-yaml`: Utilice cualquier archivo YAML existente en el directorio de instalación.
- `--use-ipv6`: Utilice IPv6 para la comunicación de Astra Trident.

logs

Puede ejecutar el `logs` Indicadores para imprimir los registros de Astra Trident.

```
Usage:
  tridentctl logs [flags]
```

Indicadores:

- `-a, --archive`: Cree un archivo de soporte con todos los registros a menos que se especifique lo contrario.
- `-h, --help`: Ayuda para registros.
- `-l, --log string`: Mostrar el registro de Astra Trident. Uno de trident|auto|trident-operator|All (valor predeterminado "auto").
- `--node string`: El nombre del nodo Kubernetes del que se van a recopilar registros del nodo pod.
- `-p, --previous`: Obtiene los registros de la instancia anterior del contenedor si existe.
- `--sidecars`: Obtener los registros de los contenedores sidecar.

send

Puede ejecutar el `send` Para enviar un recurso desde Astra Trident.

```
Usage:
  tridentctl send [option]
```

Opción disponible:
`autosupport`: Enviar un fichero AutoSupport a NetApp.

uninstall

Puede ejecutar el `uninstall` Indicadores para desinstalar Astra Trident.

```
Usage:
  tridentctl uninstall [flags]
```

Indicadores:
* `-h, --help`: Ayuda para la desinstalación.
* `--silent`: Desactiva la mayoría de la salida durante la desinstalación.

update

Puede ejecutar el `update` Comandos para modificar un recurso en Astra Trident.

```
Usage:
  tridentctl update [option]
```

Opciones disponibles:
`backend`: Actualizar un back-end en Astra Trident.

version

Puede ejecutar el `version` indicadores para imprimir la versión de `tridentctl` Y el servicio Trident que se ejecuta.

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

Indicadores:
* `--client`: Sólo versión de cliente (no se necesita ningún servidor).
* `-h, --help`: Ayuda para la versión.

Pod Security Standards (PSS) y las restricciones de contexto de seguridad (SCC)

Los estándares de seguridad de Kubernetes Pod (PSS) y las políticas de seguridad de Pod (PSP) definen los niveles de permisos y restringen el comportamiento de los POD. OpenShift Security Context restriction (SCC) define de forma similar la restricción de POD específica para OpenShift Kubernetes Engine. Para proporcionar esta personalización, Astra Trident habilita ciertos permisos durante la instalación. En las siguientes secciones se detallan los permisos establecidos por Astra Trident.

PSS reemplaza las políticas de seguridad de Pod (PSP). PSP quedó obsoleto en Kubernetes v1.21 y se eliminará en la versión 1.25. Para obtener más información, consulte "Kubernetes: Seguridad".

Contexto de Kubernetes Security y campos relacionados necesarios

Permiso	Descripción
Privilegiado	CSI requiere que los puntos de montaje sean bidireccionales, lo que significa que el receptáculo del nodo Trident debe ejecutar un contenedor privilegiado. Para obtener más información, consulte "Kubernetes: Propagación de montaje".
Conexión a redes del host	Necesario para el daemon de iSCSI. `iscsiadm` Gestiona los montajes iSCSI y utiliza la conexión a redes host para comunicarse con el daemon iSCSI.
IPC de host	NFS utiliza la comunicación entre procesos (IPC) para comunicarse con NFSD.
PID del host	Necesario para comenzar `rpc-statd` Para NFS. Astra Trident consulta los procesos de host para determinar si `rpc-statd` Se ejecuta antes de montar volúmenes NFS.

Funcionalidades	La `SYS_ADMIN` la capacidad se proporciona como parte de las capacidades predeterminadas para los contenedores con privilegios. Por ejemplo, Docker establece estas funcionalidades para los contenedores con privilegios:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Seccomp Profile siempre es "no confinado" en contenedores con privilegios; por lo tanto, no se puede activar en Astra Trident.

Permiso	Descripción
SELinux	En OpenShift, los contenedores con privilegios se ejecutan en `spc_t` El dominio ("contenedor superprivilegiado") y los contenedores sin privilegios se ejecutan en el `container_t` dominio. Encendido `containerd`, con `container-selinux` instalado, todos los contenedores se ejecutan en el `spc_t` Dominio, que desactiva SELinux de forma efectiva. Por lo tanto, Astra Trident no añade `seLinuxOptions` a los contenedores.
DAC	Los contenedores con privilegios deben ejecutarse como root. Los contenedores no privilegiados se ejecutan como root para acceder a los sockets unix necesarios para CSI.

Estándares de seguridad para POD (PSS)

Etiqueta	Descripción	Predeterminado

`pod-security.kubernetes.io/enforce`
 	
`pod-security.kubernetes.io/enforce-version`
	Permite admitir la controladora Trident y los nodos en el espacio de nombres de instalación.
	
	No cambie la etiqueta de espacio de nombres.
		`enforce: privileged`
		
		`enforce-version: <version of the current cluster or highest version of PSS tested.>`

El cambio de las etiquetas del espacio de nombres puede provocar que los POD no se programen, un "error al crear: …" O bien, "Advertencia: trident-csi-…". Si esto sucede, compruebe si la etiqueta de espacio de nombres para `privileged` se ha cambiado. En ese caso, vuelva a instalar Trident.

Directivas de seguridad de POD (PSP)

Campo	Descripción	Predeterminado
`allowPrivilegeEscalation`	Los contenedores con privilegios deben permitir la escala de privilegios.	`true`
`allowedCSIDrivers`	Trident no utiliza volúmenes efímeros de CSI en línea.	Vacío
`allowedCapabilities`	Los contenedores Trident no con privilegios no requieren más funcionalidades de las que se establece de forma predeterminada y se conceden todas las funcionalidades posibles a los contenedores con privilegios.	Vacío

Campo	Descripción	Predeterminado
`allowedFlexVolumes`	Trident no utiliza "Controlador FlexVolume", por lo tanto, no se incluyen en la lista de volúmenes permitidos.	Vacío
`allowedHostPaths`	El pod del nodo Trident monta el sistema de archivos raíz del nodo, por lo que no hay ninguna ventaja para configurar esta lista.	Vacío
`allowedProcMountTypes`	Trident no utiliza ninguna `ProcMountTypes`.	Vacío
`allowedUnsafeSysctls`	Trident no requiere que no sea seguro `sysctls`.	Vacío
`defaultAddCapabilities`	No es necesario añadir capacidades a contenedores con privilegios.	Vacío
`defaultAllowPrivilegeEscalation`	En cada POD de Trident, se permite el escalado de privilegios.	`false`
`forbiddenSysctls`	No `sysctls` se permiten.	Vacío
`fsGroup`	Los contenedores Trident se ejecutan como raíz.	`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` está ejecutándose en el nodo.	`true`
`hostPorts`	Trident no utiliza puertos de host.	Vacío
`privileged`	Los pods de nodo Trident deben ejecutar un contenedor privilegiado para montar volúmenes.	`true`
`readOnlyRootFilesystem`	Los contenedores de nodos Trident deben escribir en el sistema de archivos del nodo.	`false`
`requiredDropCapabilities`	Los pods de nodo de Trident ejecutan un contenedor privilegiado y no pueden soltar las funcionalidades.	`none`
`runAsGroup`	Los contenedores Trident se ejecutan como raíz.	`RunAsAny`

Campo	Descripción	Predeterminado
`runAsUser`	Los contenedores Trident se ejecutan como raíz.	`runAsAny`
`runtimeClass`	Trident no utiliza `RuntimeClasses`.	Vacío
`seLinux`	Trident no está configurado `seLinuxOptions` Debido a que actualmente existen diferencias en el modo en que los tiempos de ejecución de contenedores y las distribuciones de Kubernetes se encargan de SELinux.	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como raíz.	`RunAsAny`
`volumes`	Los pods de Trident requieren estos complementos de volumen.	`hostPath, projected, emptyDir`

Restricciones de contexto de seguridad (SCC)

Etiquetas	Descripción	Predeterminado
`allowHostDirVolumePlugin`	Los contenedores de nodos 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` está ejecutándose en el nodo.	`true`
`allowHostPorts`	Trident no utiliza puertos de host.	`false`
`allowPrivilegeEscalation`	Los contenedores con privilegios deben permitir la escala de privilegios.	`true`
`allowPrivilegedContainer`	Los pods de nodo Trident deben ejecutar un contenedor privilegiado para montar volúmenes.	`true`
`allowedUnsafeSysctls`	Trident no requiere que no sea seguro `sysctls`.	`none`

Etiquetas	Descripción	Predeterminado
`allowedCapabilities`	Los contenedores Trident no con privilegios no requieren más funcionalidades de las que se establece de forma predeterminada y se conceden todas las funcionalidades posibles a los contenedores con privilegios.	Vacío
`defaultAddCapabilities`	No es necesario añadir capacidades a contenedores con privilegios.	Vacío
`fsGroup`	Los contenedores Trident se ejecutan como raíz.	`RunAsAny`
`groups`	Este SCC es específico de Trident y está vinculado a su usuario.	Vacío
`readOnlyRootFilesystem`	Los contenedores de nodos Trident deben escribir en el sistema de archivos del nodo.	`false`
`requiredDropCapabilities`	Los pods de nodo de Trident ejecutan un contenedor privilegiado y no pueden soltar las funcionalidades.	`none`
`runAsUser`	Los contenedores Trident se ejecutan como raíz.	`RunAsAny`
`seLinuxContext`	Trident no está configurado `seLinuxOptions` Debido a que actualmente existen diferencias en el modo en que los tiempos de ejecución de contenedores y las distribuciones de Kubernetes se encargan de SELinux.	Vacío
`seccompProfiles`	Los contenedores privilegiados siempre funcionan "sin confinar".	Vacío
`supplementalGroups`	Los contenedores Trident se ejecutan como raíz.	`RunAsAny`
`users`	Se proporciona una entrada para vincular este SCC al usuario Trident en el espacio de nombres Trident.	n.a.
`volumes`	Los pods de Trident requieren estos complementos de volumen.	`hostPath, downwardAPI, projected, emptyDir`

Información de copyright

Información de la marca comercial