Administra y supervisa Trident

Trident

NetApp
July 01, 2026

Tabla de contenidos

Administra y supervisa Trident

Actualizar Trident

Actualizar Trident

A partir de la versión 24.02, Trident sigue un ritmo de lanzamiento de cuatro meses, entregando tres versiones principales cada año calendario. Cada nueva versión se basa en las versiones anteriores y ofrece nuevas funciones, mejoras de rendimiento, correcciones de errores y mejoras. Te recomendamos actualizar al menos una vez al año para aprovechar las nuevas funciones de Trident.

Consideraciones antes de actualizar

Al actualizar a la última versión de Trident, ten en cuenta lo siguiente:

- Solo debe haber una instancia de Trident instalada en todos los namespaces en un clúster de Kubernetes dado.
- Trident 23.07 y versiones posteriores requieren instantáneas de volumen v1 y ya no son compatibles con instantáneas alfa o beta.
- Al actualizar, es importante que proporciones `parameter.fsType` en `StorageClasses` usados por Trident. Puedes eliminar y volver a crear `StorageClasses` sin interrumpir los volúmenes preexistentes.
  - Este es un **requisito** para aplicar "contextos de seguridad" a los volúmenes SAN.
  - El directorio sample input contiene ejemplos, como `storage-class-basic.yaml.templ` y `storage-class-bronze-default.yaml`.
  - Para obtener más información, consulta "Problemas conocidos".

Paso 1: selecciona una versión

Las versiones de Trident siguen una `YY.MM` convención de nomenclatura basada en la fecha, donde "YY" son los dos últimos dígitos del año y "MM" es el mes. Las versiones de punto siguen una `YY.MM.X` convención, donde "X" es el nivel de parche. Vas a seleccionar la versión a la que quieres actualizar según la versión desde la que estás actualizando.

- Puedes hacer una actualización directa a cualquier versión de destino que esté dentro de una ventana de cuatro versiones respecto a la versión que tienes instalada. Por ejemplo, puedes actualizar directamente de 24.06 (o cualquier versión puntual de 24.06) a 25.06.
- Si estás actualizando desde una versión fuera de la ventana de cuatro versiones, realiza una actualización en varios pasos. Usa las instrucciones de actualización para la "versión anterior" de la que estás actualizando para pasar a la versión más reciente que entre en el periodo de cuatro versiones. Por ejemplo, si estás ejecutando 23.07 y quieres actualizar a 25.06:
  a. Primera actualización de 23.07 a 24.06.
  b. Luego actualiza de 24.06 a 25.06.

Cuando actualices usando el operador Trident en OpenShift Container Platform, deberías actualizar a Trident 21.01.1 o una versión posterior. El operador Trident lanzado con 21.01.0 contiene un problema conocido que se ha solucionado en 21.01.1. Para más detalles, consulta el "detalles del problema en GitHub".

Paso 2: determina el método de instalación original

Para determinar qué versión usaste para instalar Trident originalmente:

1. Usa `kubectl get pods -n trident` para examinar los pods.
   - Si no hay ningún pod de operador, Trident se instaló usando `tridentctl`.
   - Si hay un pod de operador, Trident se instaló usando el operador Trident, ya sea manualmente o usando Helm.
2. Si hay un pod de operador, usa `kubectl describe torc` para determinar si Trident se instaló usando Helm.
   - Si hay una etiqueta Helm, Trident se instaló usando Helm.
   - Si no hay etiqueta Helm, Trident se instaló manualmente usando el operador Trident.

Paso 3: selecciona un método de actualización

Por lo general, debes actualizar utilizando el mismo método que usaste para la instalación inicial, pero también puedes "moverse entre métodos de instalación". Hay dos opciones para actualizar Trident.

- "Actualiza usando el operador Trident"

Te sugerimos que revises "Entiende el flujo de trabajo de actualización del operador" antes de actualizar con el operador.

*

Actualiza con el operador

Entiende el flujo de trabajo de actualización del operador

Antes de usar el operador Trident para actualizar Trident, debes entender los procesos en segundo plano que ocurren durante la actualización. Esto incluye cambios en el controlador Trident, el Pod de controlador y los Pods de nodo, y el nodo DaemonSet que permiten actualizaciones continuas.

Gestión de la actualización del operador Trident

Uno de los muchos "ventajas de usar el operador Trident" para instalar y actualizar Trident es el manejo automático de objetos de Trident y Kubernetes sin interrumpir los volúmenes montados existentes. De esta manera, Trident puede soportar actualizaciones con cero tiempo de inactividad o "actualizaciones continuas". En concreto, el operador de Trident se comunica con el clúster de Kubernetes para:

- Elimina y vuelve a crear el deployment del controlador Trident y el DaemonSet de nodo.
- Reemplaza el Trident Controller Pod y los Trident Node Pods por nuevas versiones.

- Si un nodo no se actualiza, no impide que los nodos restantes se actualicen.
- Solo los nodos con un Trident Node Pod en ejecución pueden montar volúmenes.

Para más información sobre la arquitectura de Trident en el clúster de Kubernetes, consulta "Arquitectura de Trident".

Flujo de trabajo de actualización de Operator

Cuando inicias una actualización usando el operador Trident:

1. El **Trident operator**:
   a. Detecta la versión de Trident que tienes instalada actualmente (versión *n*).
   b. Actualiza todos los objetos de Kubernetes, incluidos CRDs, RBAC y Trident SVC.
   c. Elimina la implementación del controlador Trident para la versión *n*.
   d. Crea el despliegue del controlador Trident para la versión *n+1*.
2. **Kubernetes** crea Trident Controller Pod para *n+1*.
3. El **Trident operator**:
   a. Elimina el DaemonSet de nodo de Trident para *n*. El operador no espera a que termine el Pod de nodo.
   b. Crea el Daemonset del nodo Trident para *n+1*.
4. **Kubernetes** crea Trident Node Pods en nodos que no están ejecutando Trident Node Pod *n*. Esto garantiza que nunca haya más de un Trident Node Pod, de cualquier versión, en un nodo.

Actualiza una instalación de Trident usando el operador Trident o Helm

Puedes actualizar Trident usando el operador Trident, ya sea manualmente o usando Helm. Puedes actualizar desde una instalación del operador Trident a otra instalación del operador Trident, o actualizar desde una instalación de `tridentctl` a una versión del operador Trident. Revisa "Selecciona un método de actualización" antes de actualizar una instalación del operador Trident.

Actualizar una instalación manual

Puedes actualizar desde una instalación de operador Trident con alcance de clúster a otra instalación de operador Trident con alcance de clúster. Todas las versiones de Trident usan un operador con alcance de clúster.

Para actualizar desde Trident que se instaló utilizando el operador de espacio de nombres (versiones 20.07 a 20.10), usa las instrucciones de actualización para "tu versión instalada" de Trident.

Acerca de esta tarea

Trident proporciona un archivo de paquete que puedes usar para instalar el operador y crear objetos asociados para tu versión de Kubernetes.

- Para clústeres que ejecutan Kubernetes 1.25 o posterior, usa "bundle_post_1_25.yaml".

Antes de empezar

Asegúrate de estar usando un clúster de Kubernetes que esté ejecutando "una versión compatible de Kubernetes".

Pasos

1. Verifica tu versión de Trident:

```
./tridentctl -n trident version
```

2. Actualiza el `operator.yaml`, `tridentorchestrator_cr.yaml` y `post_1_25_bundle.yaml` con el registro y las rutas de imagen para la versión a la que estás actualizando (por ejemplo, 25.06), y el secreto correcto.
3. Elimina el operador Trident que se usó para instalar la instancia actual de Trident. Por ejemplo, si estás actualizando desde 25.02, ejecuta el siguiente comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Si personalizaste tu instalación inicial usando `TridentOrchestrator` atributos, puedes editar el objeto `TridentOrchestrator` para modificar los parámetros de instalación. Esto puede incluir cambios para especificar registros de imágenes de Trident y CSI reflejados para el modo sin conexión, habilitar registros de depuración o especificar secretos de extracción de imágenes.
5. Instala Trident usando el archivo YAML del paquete correcto para tu entorno, donde *<bundle.yaml>* es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes. Por ejemplo, si estás instalando Trident 25.06.0, ejecuta el siguiente comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Edita el trident torc para incluir la imagen 25.06.0.

Actualizar una instalación de Helm

Puedes actualizar una instalación de Trident Helm.

> Al actualizar un clúster de Kubernetes de 1.24 a 1.25 o posterior que tenga Trident instalado, debes actualizar values.yaml para establecer `excludePodSecurityPolicy` en `true` o agregar `--set excludePodSecurityPolicy=true` al comando `helm upgrade` antes de poder actualizar el clúster.

Si ya actualizaste tu clúster de Kubernetes de la versión 1.24 a la 1.25 sin actualizar el helm de Trident, la actualización de helm fallará. Para que la actualización de helm funcione, realiza estos pasos como requisitos previos:

1. Instala el complemento helm-mapkubeapis desde https://github.com/helm/helm-mapkubeapis.
2. Haz una prueba en seco de la versión de Trident en el espacio de nombres donde está instalado. Esto muestra los recursos que se limpiarán.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Realiza una ejecución completa con helm para hacer la limpieza.

```
helm mapkubeapis trident  --namespace trident
```

Pasos

1. Si "instalé Trident usando Helm", puedes usar `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0` para actualizar en un solo paso. Si no agregaste el repositorio de Helm o no puedes usarlo para actualizar:
   a. Descarga la última versión de Trident desde "la sección *Assets* en GitHub".
   b. Usa el `helm upgrade` comando donde `trident-operator-26.02.0.tgz` refleja la versión a la que quieres actualizar.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

> Si configuras opciones personalizadas durante la instalación inicial (como especificar registros privados o reflejados para las imágenes de Trident y CSI), agrega el `helm upgrade` comando usando `--set` para asegurarte de que esas opciones estén incluidas en el comando de actualización, de lo contrario los valores se restablecerán a los predeterminados.

2. Ejecuta `helm list` para verificar que tanto el chart como la versión de la app se hayan actualizado. Ejecuta `tridentctl logs` para revisar cualquier mensaje de depuración.

Actualiza desde una `tridentctl` instalación a Trident operator

Puedes actualizar a la última versión del operador Trident desde una `tridentctl` instalación. Los backends y PVC existentes estarán disponibles automáticamente.

> Antes de cambiar entre métodos de instalación, revisa "Cambiar entre métodos de instalación".

Pasos

1. Descarga la última versión de Trident.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. Crea el `tridentorchestrator` CRD a partir del manifiesto.

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

3. Implementa el operador con ámbito de clúster en el mismo espacio de nombres.

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Crea un `TridentOrchestrator`CR para instalar Trident.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirma que Trident se actualizó a la versión prevista.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

Actualiza con tridentctl

Puedes actualizar fácilmente una instalación existente de Trident usando `tridentctl`.

Acerca de esta tarea

Desinstalar y volver a instalar Trident actúa como una actualización. Cuando desinstalas Trident, la Persistent Volume Claim (PVC) y el Persistent Volume (PV) usados por el deployment de Trident no se eliminan. Los PV que ya se hayan aprovisionado seguirán disponibles mientras Trident esté fuera de línea, y Trident aprovisionará volúmenes para cualquier PVC que se cree en ese tiempo después de que vuelva a estar en línea.

Antes de empezar

Revisa "Selecciona un método de actualización" antes de actualizar usando `tridentctl`.

Pasos

1. Ejecuta el comando de desinstalación en `tridentctl` para eliminar todos los recursos asociados con Trident excepto los CRDs y los objetos relacionados.

```
./tridentctl uninstall -n <namespace>
```

2. Vuelve a instalar Trident. Consulta "Instala Trident usando tridentctl".

> No interrumpas el proceso de actualización. Asegúrate de que el instalador se ejecute hasta el final.

Administra Trident usando tridentctl

El "Paquete de instalación de Trident" incluye la `tridentctl` utilidad de línea de comandos para proporcionar un acceso sencillo a Trident. Los usuarios de Kubernetes con privilegios suficientes pueden usarla para instalar Trident o gestionar el namespace que contiene el pod de Trident.

Comandos y banderas globales

Puedes ejecutar `tridentctl help` para obtener una lista de comandos disponibles para `tridentctl` o añadir la bandera `--help` a cualquier comando para ver una lista de opciones y banderas para ese comando específico.

```
tridentctl [command] [--optional-flag]
```

La utilidad Trident `tridentctl` admite los siguientes comandos y banderas globales.

Comandos

`create`
Agrega un recurso a Trident.

`delete`
Elimina uno o varios recursos de Trident.

`get`
Obtén uno o más recursos de Trident.

`help`
Ayuda sobre cualquier comando.

`images`
Imprime una tabla de las imágenes de contenedores que necesita Trident.

`import`
Importa un recurso existente a Trident.

`install`
Instala Trident.

`logs`
Imprime los registros de Trident.

`send`
Envía un recurso desde Trident.

`uninstall`
Desinstala Trident.

`update`
Modifica un recurso en Trident.

`update backend state`
Suspende temporalmente las operaciones de backend.

`upgrade`
Actualiza un recurso en Trident.

`version`
Imprime la versión de Trident.

Banderas globales

`-d, --debug`
Salida de depuración.

`-h, --help`
Ayuda para `tridentctl`.

`-k, --kubeconfig string`
Especifica la `KUBECONFIG` ruta para ejecutar comandos localmente o de un clúster de Kubernetes a otro.

> Como alternativa, puedes exportar la `KUBECONFIG` variable para que apunte a un clúster de Kubernetes específico y emitir `tridentctl` comandos a ese clúster.

`-n, --namespace string`
Espacio de nombres de la implementación de Trident.

`-o, --output string`
Formato de salida. Uno de json|yaml|name|wide|ps (predeterminado).

`-s, --server string`
Dirección/puerto de la interfaz de REST de Trident.

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

Opciones de comando y flags

crear

Usa el `create` comando para agregar un recurso a Trident.

```
tridentctl create [option]
```

Opciones

`backend`: agrega un backend a Trident.

borrar

Usa el comando `delete` para eliminar uno o más recursos de Trident.

```
tridentctl delete [option]
```

Opciones

`backend`: eliminar uno o más backends de almacenamiento de Trident.
`snapshot`: eliminar una o más instantáneas de volumen de Trident.
`storageclass`: eliminar una o más clases de almacenamiento de Trident.

`volume`: eliminar uno o más volúmenes de almacenamiento de Trident.

obtener

Usa el `get` comando para obtener uno o más recursos de Trident.

```
tridentctl get [option]
```

Opciones

`backend`: Obtén uno o más backends de almacenamiento de Trident.
`snapshot`: Obtén una o más instantáneas de Trident.
`storageclass`: Obtén una o más clases de almacenamiento de Trident.
`volume`: Obtén uno o más volúmenes de Trident.

Banderas

`-h`, `--help`: Ayuda para volúmenes.
`--parentOfSubordinate string`: Limitar consulta a volumen de origen subordinado.
`--subordinateOf string`: Limitar consulta a subordinados de volumen.

imágenes

Usa `images`banderas para imprimir una tabla de las imágenes de contenedores que necesita Trident.

```
tridentctl images [flags]
```

Banderas

`-h`, `--help`: ayuda para imágenes.
`-v`, `--k8s-version string`: versión semántica del clúster de Kubernetes.

importar volumen

Usa el `import volume` comando para importar un volumen existente a Trident.

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

Alias

`volume`, `v`

Banderas

`-f`, `--filename string`: ruta al archivo YAML o JSON PVC.
`-h`, `--help`: ayuda para el volumen.
`--no-manage`: crear solo PV/PVC. No asumas la gestión del ciclo de vida del volumen.

instalar

Usa las `install` banderas para instalar Trident.

```
tridentctl install [flags]
```

Banderas

`--autosupport-image string`: La imagen del contenedor para Autosupport Telemetry

(predeterminado "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: La dirección/puerto de un proxy para enviar Autosupport Telemetry.
`--enable-node-prep`: Intentar instalar los paquetes requeridos en los nodos.
`--generate-custom-yaml`: Generar archivos YAML sin instalar nada.
`-h, --help`: Ayuda para instalar.
`--http-request-timeout`: Anular el tiempo de espera de la solicitud HTTP para la API de REST del controlador Trident (predeterminado 1m30s).
`--image-registry string`: La dirección/puerto de un registro interno de imágenes.
`--k8s-timeout duration`: El tiempo de espera para todas las operaciones de Kubernetes (predeterminado 3m0s).
`--kubelet-dir string`: La ubicación en el host del estado interno de kubelet (predeterminado "/var/lib/kubelet").
`--log-format string`: El formato de registro de Trident (text, json) (predeterminado "text").
`--node-prep`: Permite que Trident prepare los nodos del clúster de Kubernetes para gestionar volúmenes usando el protocolo de almacenamiento de datos especificado. **Actualmente, `iscsi` es el único valor admitido. Beginning with OpenShift 4.19, la versión mínima de Trident compatible con esta función es 25.06.1.**
`--pv string`: El nombre del PV heredado que usa Trident, asegúrate de que no exista (predeterminado "trident").
`--pvc string`: El nombre del PVC heredado que usa Trident, asegúrate de que no exista (predeterminado "trident").
`--silence-autosupport`: No enviar paquetes de autosupport a NetApp automáticamente (predeterminado true).
`--silent`: Deshabilitar la mayoría de la salida durante la instalación.
`--trident-image string`: La imagen de Trident que se va a instalar.
`--k8s-api-qps`: El límite de consultas por segundo (QPS) para las solicitudes de la API de Kubernetes (predeterminado 100; opcional).
`--use-custom-yaml`: Usar cualquier archivo YAML existente que haya en el directorio de setup.
`--use-ipv6`: Usar IPv6 para la comunicación de Trident.

registros

Usa `logs`banderas para imprimir los registros de Trident.

```
tridentctl logs [flags]
```

Banderas

`-a, --archive`: Crea un archivo de soporte con todos los registros a menos que se especifique lo contrario.
`-h, --help`: Ayuda para registros.
`-l, --log string`: Registro de Trident para mostrar. Uno de trident|auto|trident-operator|all (predeterminado "auto").
`--node string`: El nombre de nodo de Kubernetes del que se recopilarán los registros del pod del nodo.
`-p, --previous`: Obtén los registros de la instancia anterior del contenedor si existe.
`--sidecars`: Obtén los registros de los contenedores sidecar.

enviar

Usa el `send` comando para enviar un recurso desde Trident.

```
tridentctl send [option]
```

Opciones

`autosupport`: Envía un archivo de Autosupport a NetApp.

desinstalar

Usa `uninstall` flags para desinstalar Trident.

```
tridentctl uninstall [flags]
```

Banderas

`-h, --help`: ayuda para la desinstalación.
`--silent`: deshabilitar la mayoría de las salidas durante la desinstalación.

actualizar

Usa el `update` comando para modificar un recurso en Trident.

```
tridentctl update [option]
```

Opciones

`backend`: actualiza un backend en Trident.

actualiza el estado del backend

Usa el `update backend state` comando para suspender o reanudar las operaciones del backend.

```
tridentctl update backend state <backend-name> [flag]
```

Puntos a considerar

- Si se crea un backend utilizando un TridentBackendConfig (tbc), el backend no se puede actualizar usando un archivo `backend.json`.
- Si el `userState` se ha establecido en un tbc, no se puede modificar usando el comando `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Para recuperar la capacidad de configurar el `userState` mediante tridentctl después de que se haya configurado mediante tbc, el campo `userState` debe eliminarse de tbc. Esto se puede hacer usando el comando `kubectl edit tbc`. Después de eliminar el campo `userState`, puedes usar el comando `tridentctl update backend state` para cambiar el `userState` de un backend.
- Usa el `tridentctl update backend state` para cambiar el `userState`. También puedes actualizar el `userState` usando `TridentBackendConfig` o `backend.json` archivo; esto activa una reinicialización completa del backend y puede llevar mucho tiempo.

  #### Banderas

  `-h, --help`: Ayuda para el estado del backend.
  `--user-state`: Configura en `suspended` para pausar las operaciones del backend. Configura en `normal` para reanudar las operaciones del backend. Cuando se configura en `suspended`:

- `AddVolume` y `Import Volume` están en pausa.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`,

`ReconcileNodeAccess` siguen estando disponibles.

También puedes actualizar el estado del backend usando `userState` el campo en el archivo de configuración del backend `TridentBackendConfig` o `backend.json`. Para más información, consulta "Opciones para gestionar backends" y "Realiza la gestión del backend con kubectl".

Ejemplo:

JSON

Sigue estos pasos para actualizar el `userState` usando el `backend.json` archivo:

1. Edita el `backend.json` archivo para incluir el `userState` campo con su valor establecido en 'suspended'.
2. Actualiza el backend usando el `tridentctl update backend` comando y la ruta al archivo `backend.json` actualizado.

   Ejemplo: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Puedes editar el tbc después de aplicarlo usando el `kubectl edit <tbc-name> -n <namespace>` comando. El siguiente ejemplo actualiza el estado del backend a suspendido usando la opción `userState: suspended`:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

versión

Usa `version` banderas para imprimir la versión de `tridentctl` y el servicio Trident en ejecución.

```
tridentctl version [flags]
```

Banderas

`--client`: Solo versión de cliente (no requiere servidor).
`-h, --help`: Ayuda para la versión.

Compatibilidad con complementos

Tridentctl admite plugins similares a kubectl. Tridentctl detecta un plugin si el nombre del archivo binario del plugin sigue el esquema "tridentctl-<plugin>" y el binario está ubicado en una carpeta listada en la variable de entorno PATH. Todos los plugins detectados se muestran en la sección de plugins de la ayuda de tridentctl. Opcionalmente, también puedes limitar la búsqueda especificando una carpeta de plugins en la variable de entorno TRIDENTCTL_PLUGIN_PATH (ejemplo: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Si se usa la variable, tridentctl busca solo en la carpeta especificada.

Monitorea Trident

Trident proporciona un conjunto de puntos finales de métricas de Prometheus que puedes usar para monitorear el rendimiento de Trident.

Descripción general

Las métricas proporcionadas por Trident te permiten hacer lo siguiente:

- Monitorea el estado y la configuración de Trident. Puedes examinar qué tan exitosas son las operaciones y si puede comunicarse con los backends como se espera.
- Examina la información de uso del backend y entiende cuántos volúmenes se aprovisionan en un backend y la cantidad de espacio consumido, y así sucesivamente.
- Mantén un mapeo de la cantidad de volúmenes aprovisionados en los backends disponibles.
- Monitorea el rendimiento. Puedes ver cuánto tarda Trident en comunicarse con los backends y realizar operaciones.

> ⓘ De forma predeterminada, las métricas de Trident se exponen en el puerto de destino `8001` en el endpoint `/metrics`. Estas métricas están **habilitadas por defecto** cuando instalas Trident. También puedes configurar el consumo de métricas de Trident a través de HTTPS en el puerto `8444`.

Lo que necesitas

- Un clúster de Kubernetes con Trident instalado.
- Una instancia de Prometheus. Esto puede ser un "Implementación de Prometheus en contenedores" o puedes elegir ejecutar Prometheus como un "aplicación nativa".

Paso 1: define un objetivo de Prometheus

Deberías definir un objetivo de Prometheus para recopilar las métricas y obtener información sobre los

backends que Trident administra, los volúmenes que crea y así sucesivamente. Consulta "Documentación de Prometheus Operator".

Paso 2: Crea un ServiceMonitor de Prometheus

Para consumir las métricas de Trident, debes crear un Prometheus ServiceMonitor que supervise el `trident-csi` servicio y escuche en el `metrics` puerto. Un ejemplo de ServiceMonitor se ve así:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Esta definición de ServiceMonitor recupera las métricas devueltas por el `trident-csi` servicio y busca específicamente el `metrics` endpoint del servicio. Como resultado, Prometheus ahora está configurado para entender las métricas de Trident.

Además de las métricas disponibles directamente desde Trident, kubelet expone muchas `kubelet_volume_*` métricas a través de su propio endpoint de métricas. Kubelet puede proporcionar información sobre los volúmenes que están conectados, los pods y otras operaciones internas que maneja. Consulta "aquí".

Consume métricas de Trident a través de HTTPS

Para consumir métricas de Trident a través de HTTPS (puerto 8444), debes modificar la definición de ServiceMonitor para incluir la configuración TLS. También necesitas copiar el `trident-csi` secreto desde el `trident` namespace al namespace donde se está ejecutando Prometheus. Puedes hacer esto usando el siguiente comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un ejemplo de ServiceMonitor para métricas HTTPS se ve así:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident admite métricas HTTPS en todos los métodos de instalación: tridentctl, Helm chart y Operator:

- Si estás usando el `tridentctl install` comando, puedes pasar la `--https-metrics` bandera para habilitar las métricas HTTPS.
- Si estás usando el gráfico Helm, puedes configurar el `httpsMetrics` parámetro para habilitar las métricas HTTPS.
- Si estás usando archivos YAML, puedes agregar la `--https_metrics` bandera al `trident-main` contenedor en el `trident-deployment.yaml` archivo.

Paso 3: Consultar métricas de Trident con PromQL

PromQL es bueno para crear expresiones que devuelven series de tiempo o datos tabulares.

Aquí hay algunas consultas PromQL que puedes usar:

Obtén información de salud de Trident

- **Porcentaje de respuestas HTTP 2XX de Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Porcentaje de respuestas REST de Trident mediante código de estado**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duración media en ms de las operaciones realizadas por Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtén información sobre el uso de Trident

- **Tamaño promedio del volumen**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espacio total de volumen aprovisionado por cada backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtén el uso del volumen individual

> Esto solo está habilitado si también se recopilan métricas de kubelet.

- **Porcentaje de espacio utilizado para cada volumen**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Conoce la telemetría de Trident AutoSupport

De forma predeterminada, Trident envía métricas de Prometheus e información básica del backend a NetApp cada día.

- Para evitar que Trident envíe métricas de Prometheus e información básica del backend a NetApp, pasa la `--silence-autosupport` bandera durante la instalación de Trident.
- Trident también puede enviar registros de contenedores a NetApp Support bajo demanda mediante `tridentctl send autosupport`. Necesitarás activar Trident para que suba sus registros. Antes de enviar los registros, deberías aceptar la NetApp "política de privacidad".
- A menos que se especifique, Trident recupera los registros de las últimas 24 horas.
- Puedes especificar el periodo de retención de registros con la `--since` bandera. Por ejemplo: `tridentctl send autosupport --since=1h`. Esta información se recopila y se envía mediante un `trident-autosupport` contenedor que se instala junto con Trident. Puedes obtener la imagen del contenedor en "Trident AutoSupport".
- Trident AutoSupport no recopila ni transmite información personal identificable (PII) ni información personal. Viene con una "CLUF" que no aplica a la propia imagen del contenedor de Trident. Puedes aprender más sobre el compromiso de NetApp con la seguridad y la confianza de los datos "aquí".

Un ejemplo de carga útil enviada por Trident se ve así:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- Los mensajes de AutoSupport se envían al endpoint de AutoSupport de NetApp. Si estás usando un registro privado para almacenar imágenes de contenedor, puedes usar la bandera `--image-registry`.
- También puedes configurar las URL de proxy generando los archivos YAML de instalación. Esto se puede hacer usando `tridentctl install --generate-custom-yaml` para crear los archivos YAML y añadiendo el argumento `--proxy-url` para el contenedor `trident-autosupport` en `trident-`

`deployment.yaml`.

Desactivar métricas de Trident

Para **deshabilitar** que se informen las métricas, debes generar YAML personalizados (usando la `--generate-custom-yaml` bandera) y editarlos para quitar la `--metrics` bandera de ser invocada para el contenedor `trident-main`.

Desinstala Trident

Debes usar el mismo método para desinstalar Trident que usaste para instalar Trident.

Acerca de esta tarea

- Si necesitas una solución para errores observados después de una actualización, problemas de dependencias o una actualización fallida o incompleta, deberías desinstalar Trident y reinstalar la versión anterior usando las instrucciones específicas para ese "versión". Esta es la única forma recomendada de *downgrade* a una versión anterior.
- Para facilitar la actualización y reinstalación, desinstalar Trident no elimina los CRD ni los objetos relacionados creados por Trident. Si necesitas eliminar Trident por completo y todos sus datos, consulta "Eliminar por completo Trident y CRDs".

Antes de empezar

Si vas a decomisionar clústeres de Kubernetes, debes eliminar todas las aplicaciones que usan volúmenes creados por Trident antes de desinstalar. Esto asegura que los PVC se despublican en los nodos de Kubernetes antes de eliminarlos.

Determina el método de instalación original

Deberías usar el mismo método para desinstalar Trident que usaste para instalarlo. Antes de desinstalar, verifica qué versión usaste para instalar Trident originalmente.

1. Usa `kubectl get pods -n trident` para examinar los pods.
   - Si no hay ningún pod de operador, Trident se instaló usando `tridentctl`.
   - Si hay un pod de operador, Trident se instaló usando el operador Trident, ya sea manualmente o usando Helm.
2. Si hay un pod de operador, usa `kubectl describe tproc trident` para determinar si Trident se instaló usando Helm.
   - Si hay una etiqueta Helm, Trident se instaló usando Helm.
   - Si no hay etiqueta Helm, Trident se instaló manualmente usando el operador Trident.

Desinstala una instalación del operador Trident

Puedes desinstalar una instalación de operador Trident manualmente o usando Helm.

Desinstala la instalación manual

Si instalaste Trident usando el operador, puedes desinstalarlo haciendo una de las siguientes cosas:

1. **Edita `TridentOrchestrator` CR y establece la bandera de desinstalación**:

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

Cuando el `uninstall` flag se establece en `true`, el operador de Trident desinstala Trident, pero no elimina el TridentOrchestrator en sí. Deberías limpiar el TridentOrchestrator y crear uno nuevo si quieres instalar Trident otra vez.

2. **Eliminar `TridentOrchestrator`**: al eliminar el `TridentOrchestrator` CR que se utilizó para desplegar Trident, le indicas al operador que desinstale Trident. El operador procesa la eliminación de `TridentOrchestrator` y procede a eliminar el despliegue de Trident y el daemonset, borrando los pods de Trident que había creado como parte de la instalación.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Desinstalar la instalación de Helm

Si instalaste Trident usando Helm, puedes desinstalarlo usando `helm uninstall`.

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstala una `tridentctl` instalación

Usa el comando `uninstall` en `tridentctl` para eliminar todos los recursos asociados con Trident excepto los CRDs y los objetos relacionados:

```
./tridentctl uninstall -n <namespace>
```