



Instala Trident

Trident

NetApp
July 01, 2026

Tabla de contenidos

- Instala Trident 1
 - Conoce la instalación de Trident 1
 - Información crítica sobre Trident 26.02 1
 - Antes de empezar 1
 - Elige tu método de instalación 2
 - Elige tu modo de instalación 4
 - Selecciona el proceso según tu método y modo 4
 - Cambiar entre métodos de instalación 5
 - Otras opciones de configuración conocidas 5
 - Instala usando el operador Trident 5
 - Implementa manualmente el operador Trident (modo estándar) 5
 - Despliega manualmente el operador Trident (modo offline) 11
 - Implementa el operador de Trident usando Helm (modo estándar) 17
 - Implementa el operador de Trident usando Helm (modo sin conexión) 25
 - Personaliza la instalación del operador Trident 33
- Instala usando tridentctl 45
 - Instala usando tridentctl 45
 - Personaliza la instalación de tridentctl 49
- Instala usando el operador certificado de OpenShift 50
 - Instala Trident usando OpenShift OperatorHub 50
 - Cambia al operador Trident certificado OpenShift 53

Instala Trident

Conoce la instalación de Trident

Para garantizar que Trident se pueda instalar en una amplia variedad de entornos y organizaciones, NetApp ofrece múltiples opciones de instalación. Puedes instalar Trident usando el operador Trident (manualmente o usando Helm) o con `tridentctl`. Este tema te da información importante para que elijas el proceso de instalación adecuado para ti.

Información crítica sobre Trident 26.02

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.35 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Antes de empezar

Independientemente de tu ruta de instalación, debes tener:

- Privilegios completos para un clúster de Kubernetes compatible que ejecute una versión compatible de Kubernetes y con los requisitos de funciones habilitados. Revisa la ["requisitos"](#) para más detalles.
- Acceso a un sistema de almacenamiento NetApp compatible.
- Capacidad de montar volúmenes desde todos los nodos de trabajo de Kubernetes.
- Un host Linux con `kubect1` (o `oc`, si estás usando OpenShift) instalado y configurado para administrar el clúster de Kubernetes que quieres usar.
- La ``KUBECONFIG`` variable de entorno configurada para apuntar a la configuración de tu clúster de Kubernetes.
- Si estás usando Kubernetes con Docker Enterprise, ["Sigue sus pasos para habilitar el acceso CLI"](#).
- El clúster debe admitir cargas de trabajo privilegiadas.



Si aún no te has familiarizado con el ["conceptos básicos"](#), ahora es un buen momento para hacerlo.

Elige tu método de instalación

Selecciona el método de instalación que sea mejor para ti. También deberías revisar las consideraciones para ["moverse entre métodos"](#) antes de tomar tu decisión.

Usando el operador Trident

Ya sea que implementes manualmente o usando Helm, el operador Trident es una gran manera de simplificar la instalación y administrar dinámicamente los recursos de Trident. Incluso puedes ["Personaliza la implementación de tu operador Trident"](#) usando los atributos en el `TridentOrchestrator` recurso personalizado (CR).

Los beneficios de usar el operador Trident incluyen:

Creación de objetos Trident

El operador Trident crea automáticamente los siguientes objetos para tu versión de Kubernetes.

- ServiceAccount para el operador
- ClusterRole y ClusterRoleBinding a la ServiceAccount
- Dedicado PodSecurityPolicy (para Kubernetes 1.25 y anteriores)
- El propio operador

Responsabilidad de recursos

El operador Trident de ámbito de clúster administra los recursos asociados a una instalación de Trident a nivel de clúster. Esto mitiga los errores que podrían producirse al mantener recursos de ámbito de clúster usando un operador de ámbito de espacio de nombres. Esto es esencial para la reparación automática y la aplicación de parches.

Capacidad de reparación automática

El operador supervisa la instalación de Trident y toma medidas activas para solucionar problemas, como cuando se elimina la implementación o si se modifica accidentalmente. A `trident-operator-
<generated-id>` pod se crea que asocia un `TridentOrchestrator` CR con una instalación de Trident. Esto asegura que solo haya una instancia de Trident en el clúster y controla su configuración, asegurando que la instalación sea idempotente. Cuando se hacen cambios en la instalación (como eliminar la implementación o el daemonset del nodo), el operador los identifica y los arregla individualmente.

Actualizaciones fáciles de instalaciones existentes

Puedes actualizar fácilmente una implementación existente con el operador. Solo necesitas editar el `TridentOrchestrator` CR para hacer actualizaciones en una instalación.

Por ejemplo, considera un escenario donde necesitas habilitar Trident para generar registros de depuración. Para hacer esto, modifica tu `TridentOrchestrator` para establecer `spec.debug` en `true`:

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

Después de `TridentOrchestrator` actualizarse, el operador procesa las actualizaciones y aplica parches a la instalación existente. Esto podría desencadenar la creación de nuevos pods para modificar la instalación en consecuencia.

Reinstalación limpia

El operador Trident con ámbito de clúster permite la eliminación limpia de recursos con ámbito de clúster. Los usuarios pueden desinstalar Trident por completo y reinstalarlo fácilmente.

Manejo automático de actualizaciones de Kubernetes

Cuando la versión de Kubernetes del clúster se actualiza a una versión compatible, el operador actualiza automáticamente una instalación de Trident existente y la cambia para asegurarse de que cumpla con los requisitos de la versión de Kubernetes.



Si el clúster se actualiza a una versión no compatible, el operador impide instalar Trident. Si Trident ya se instaló con el operador, se muestra una advertencia para indicar que Trident está instalado en una versión de Kubernetes no compatible.

Usando `tridentctl`

Si tienes una implementación existente que necesita actualizarse o si buscas personalizar tu implementación al máximo, deberías considerar `tridentctl`. Este es el método convencional para implementar Trident.

Puedes generar los manifiestos para los recursos de Trident. Esto incluye el deployment, daemonset, service account y el cluster role que Trident crea como parte de su instalación.



A partir de la versión 22.04, las claves AES ya no se regenerarán cada vez que se instale Trident. Con esta versión, Trident instalará un nuevo objeto secreto que persiste entre instalaciones. Esto significa que, `tridentctl` en 22.04 puedes desinstalar versiones anteriores de Trident, pero las versiones anteriores no pueden desinstalar instalaciones de 22.04. Selecciona el método de instalación adecuado.

Elige tu modo de instalación

Determina tu proceso de implementación según el *modo de instalación* (Standard, Offline o Remote) que requiera tu organización.

Instalación estándar

Esta es la forma más sencilla de instalar Trident y funciona en la mayoría de los entornos que no imponen restricciones de red. El modo de instalación estándar utiliza los registros predeterminados para almacenar las imágenes requeridas de Trident (`docker.io`) y CSI (`registry.k8s.io`).

Cuando usas el modo estándar, el instalador de Trident:

- Obtiene las imágenes del contenedor a través de Internet
- Crea un deployment o un daemonset de nodo, que inicia pods de Trident en todos los nodos elegibles del clúster de Kubernetes

Instalación sin conexión

Podría requerirse el modo de instalación sin conexión en una ubicación segura o con aislamiento. En este escenario, puedes crear un único registro privado duplicado o dos registros duplicados para almacenar las imágenes de Trident y CSI necesarias.



Independientemente de la configuración de tu registro, las imágenes CSI deben residir en un solo registro.

Instalación remota

Aquí tienes una descripción a grandes rasgos del proceso de instalación remota:

- Implementa la versión adecuada de `kubectl` en la máquina remota desde donde quieres implementar Trident.
- Copia los archivos de configuración del clúster de Kubernetes y establece la variable de entorno `KUBECONFIG` en la máquina remota.
- Inicia un comando `kubectl get nodes` para verificar que puedes conectarte al clúster de Kubernetes requerido.
- Completa la implementación desde la máquina remota usando los pasos de instalación estándar.

Selecciona el proceso según tu método y modo

Después de que hayas tomado tus decisiones, selecciona el proceso apropiado.

Método	Modo de instalación
Trident operator (manualmente)	"Instalación estándar"
	"Instalación sin conexión"
Operador Trident (Helm)	"Instalación estándar"
	"Instalación sin conexión"

Método	Modo de instalación
tridentctl	"Instalación estándar o fuera de línea"

Cambiar entre métodos de instalación

Puedes decidir cambiar tu método de instalación. Antes de hacerlo, considera lo siguiente:

- Usa siempre el mismo método para instalar y desinstalar Trident. Si has implementado con `tridentctl`, deberías usar la versión adecuada del binario `tridentctl` para desinstalar Trident. De igual forma, si estás implementando con el operador, deberías editar el CR `TridentOrchestrator` y establecer `spec.uninstall=true` para desinstalar Trident.
- Si tienes una implementación basada en operador que quieres eliminar y usar en su lugar `tridentctl` para implementar Trident, primero deberías editar `TridentOrchestrator` y establecer `spec.uninstall=true` para desinstalar Trident. Luego elimina `TridentOrchestrator` y la implementación del operador. Después puedes instalar usando `tridentctl`.
- Si tienes una implementación manual basada en operadores y quieres usar la implementación del operador Trident basada en Helm, primero deberías desinstalar manualmente el operador y luego realizar la instalación con Helm. Esto permite que Helm implemente el operador Trident con las etiquetas y anotaciones requeridas. Si no haces esto, tu implementación del operador Trident basada en Helm fallará con un error de validación de etiquetas y un error de validación de anotaciones.
- Si tienes una implementación basada en `tridentctl`, puedes hacer una implementación basada en Helm o en Operator sin desinstalar Trident.

Otras opciones de configuración conocidas

Al instalar Trident en productos VMWare Tanzu Portfolio:

- El `--kubelet-dir`flag` debe establecerse en la ubicación del directorio de kubelet. Por defecto, esto es ``/var/vcap/data/kubelet`.

Especificar la ubicación de kubelet usando `--kubelet-dir` se sabe que funciona para Trident Operator, Helm y `tridentctl` implementaciones.

Instala usando el operador Trident

Implementa manualmente el operador Trident (modo estándar)

Puedes desplegar manualmente el operador Trident para instalar Trident. Este proceso aplica a instalaciones donde las imágenes de contenedor requeridas por Trident no están almacenadas en un registro privado. Si tienes un registro de imágenes privado, usa el ["proceso para el despliegue fuera de línea"](#).

Información crítica sobre Trident 26.02

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.35 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Despliega manualmente el operador Trident e instala Trident

Revisa "[la visión general de la instalación](#)" para asegurarte de que cumpliste los requisitos previos de instalación y seleccionaste la opción de instalación correcta para tu entorno.

Antes de empezar

Antes de comenzar la instalación, inicia sesión en el host Linux y verifica que está gestionando un servidor en funcionamiento, "[clúster de Kubernetes compatible](#)" y que tienes los privilegios necesarios.



Con OpenShift, usa `oc` en vez de `kubectl` en todos los ejemplos que siguen, y primero inicia sesión como **system:admin** ejecutando `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica tu versión de Kubernetes:

```
kubectl version
```

2. Verifica los privilegios del administrador del clúster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifica que puedes lanzar un pod que use una imagen de Docker Hub y acceder a tu sistema de almacenamiento a través de la red de pods:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

Paso 1: descarga el paquete de instalación de Trident

El paquete instalador de Trident contiene todo lo que necesitas para desplegar el operador Trident e instalar Trident. Descarga y extrae la última versión del instalador de Trident desde "[la sección Assets en GitHub](#)".

```
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
```

Paso 2: crea el `TridentOrchestrator`CRD

Crea la `TridentOrchestrator` Custom Resource Definition (CRD). Vas a crear un `TridentOrchestrator` Custom Resource más adelante. Usa la versión CRD YAML apropiada en `deploy/crds` para crear el `TridentOrchestrator` CRD.

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

Paso 3: despliega el operador Trident

El instalador de Trident proporciona un archivo de paquete que puede usarse para instalar el operador y crear objetos asociados. El archivo de paquete es una forma sencilla de desplegar el operador e instalar Trident usando una configuración predeterminada.

- Para los clústeres que ejecutan Kubernetes 1.24, usa `bundle_pre_1_25.yaml`.
- Para clústeres que ejecutan Kubernetes 1.25 o posterior, usa `bundle_post_1_25.yaml`.

Antes de empezar

- Por defecto, el instalador de Trident despliega el operador en el `trident` namespace. Si el `trident` namespace no existe, créalo usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para desplegar el operador en un espacio de nombres distinto del `trident` namespace, actualiza `serviceaccount.yaml`, `clusterrolebinding.yaml` y `operator.yaml` y genera tu archivo `bundle` usando el `kustomization.yaml`.
 - a. Crea el `kustomization.yaml` usando el siguiente comando donde `<bundle.yaml>` es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes.

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- b. Compila el paquete usando el siguiente comando donde `<bundle.yaml>` es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes.

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

Pasos

1. Crea los recursos y despliega el operador:

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

2. Verifica que se hayan creado el operador, el deployment y los replicaset.

```
kubectl get all -n <operator-namespace>
```



Solo debe haber **una instancia** del operador en un clúster de Kubernetes. No crees varias implementaciones del operador Trident.

Paso 4: crea el `TridentOrchestrator` e instala Trident

Ahora puedes crear el `TridentOrchestrator` e instalar Trident. Opcionalmente, puedes ["personaliza tu instalación Trident"](#) usando los atributos en la `TridentOrchestrator spec`.

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:       true
  Namespace:   trident
  nodePrep:
    - iscsi
Status:
  Current Installation Params:
    IPv6:                false
    Autosupport Hostname:
    Autosupport Image:    netapp/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:            text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:26.02.0
  Message:              Trident installed Namespace:
trident
  Status:                Installed
  Version:                v26.02.0
Events:
  Type Reason Age From Message ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

Verifica la instalación

Hay varias formas de verificar tu instalación.

Usando `TridentOrchestrator` status

El estado de `TridentOrchestrator` indica si la instalación fue exitosa y muestra la versión de Trident instalada. Durante la instalación, el estado de `TridentOrchestrator` cambia de `Installing` a `Installed`. Si observas el estado de `Failed` y el operador no puede recuperarse por sí solo, ["revisa los registros"](#).

Estado	Descripción
Instalando	El operador está instalando Trident usando este <code>TridentOrchestrator</code> CR.
Instalado	Trident se ha instalado correctamente.
Desinstalar	El operador está desinstalando Trident porque <code>spec.uninstall=true</code> .
Desinstalado	Trident está desinstalado.
Con errores	El operador no pudo instalar, parchear, actualizar o desinstalar Trident; el operador intentará automáticamente recuperarse de este estado. Si este estado persiste necesitarás hacer una solución de problemas.
Actualizando	El operador está actualizando una instalación existente.
Error	El <code>TridentOrchestrator</code> no se usa. Ya existe otro.

Uso del estado de creación de pods

Puedes confirmar si la instalación de Trident se completó revisando el estado de los pods creados:

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
trident-controller-7d466bf5c7-v4cpw	6/6	Running	0
1m			
trident-node-linux-mr6zc	2/2	Running	0
1m			
trident-node-linux-xrp7w	2/2	Running	0
1m			
trident-node-linux-zh2jt	2/2	Running	0
1m			
trident-operator-766f7b8658-ldzsv	1/1	Running	0
3m			

Usando `tridentctl`

Puedes usar `tridentctl` para comprobar la versión de Trident instalada.

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.02.0        | 26.02.0        |
+-----+-----+
```

Despliega manualmente el operador Trident (modo offline)

Puedes desplegar manualmente el operador Trident para instalar Trident. Este proceso aplica a instalaciones donde las imágenes de contenedor requeridas por Trident se almacenan en un registro privado. Si no tienes un registro de imágenes privado, usa el ["proceso para la implementación estándar"](#).

Información crítica sobre Trident

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.35 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Despliega manualmente el operador Trident e instala Trident

Revisa ["la visión general de la instalación"](#) para asegurarte de que cumpliste los requisitos previos de instalación y seleccionaste la opción de instalación correcta para tu entorno.

Antes de empezar

Inicia sesión en el host Linux y verifica que está gestionando un servidor que funciona y ["clúster de Kubernetes compatible"](#) y que tienes los privilegios necesarios.



Con OpenShift, usa `oc` en vez de `kubectl` en todos los ejemplos que siguen, y primero inicia sesión como **system:admin** ejecutando `oc login -u system:admin` o `oc login -u kube-admin`.

1. Verifica tu versión de Kubernetes:

```
kubectl version
```

2. Verifica los privilegios del administrador del clúster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifica que puedes lanzar un pod que use una imagen de Docker Hub y acceder a tu sistema de almacenamiento a través de la red de pods:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

Paso 1: descarga el paquete de instalación de Trident

El paquete instalador de Trident contiene todo lo que necesitas para desplegar el operador Trident e instalar Trident. Descarga y extrae la última versión del instalador de Trident desde ["la sección Assets en GitHub"](#).

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Paso 2: crea el `TridentOrchestrator`CRD

Crea la `TridentOrchestrator` Custom Resource Definition (CRD). Vas a crear una `TridentOrchestrator` Custom Resources más adelante. Usa la versión adecuada de CRD YAML en `deploy/crds` para crear la `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Paso 3: actualiza la ubicación del registro en el operador

En `/deploy/operator.yaml`, actualiza `image: docker.io/netapp/trident-operator:26.02.0` para reflejar la ubicación de tu registro de imágenes. Tu ["Imágenes de Trident y CSI"](#) puede estar en un registro o en registros diferentes, pero todas las imágenes CSI deben estar en el mismo registro. Por ejemplo:

- `image: <your-registry>/trident-operator:26.02.0` si tus imágenes están todas ubicadas en un registro.
- `image: <your-registry>/netapp/trident-operator:26.02.0` si tu imagen de Trident está en un registro diferente al de tus imágenes CSI.

Paso 4: despliega el operador Trident

El instalador de Trident proporciona un archivo de paquete que puede usarse para instalar el operador y crear objetos asociados. El archivo de paquete es una forma sencilla de desplegar el operador e instalar Trident usando una configuración predeterminada.

- Para los clústeres que ejecutan Kubernetes 1.24, usa `bundle_pre_1_25.yaml`.
- Para clústeres que ejecutan Kubernetes 1.25 o posterior, usa `bundle_post_1_25.yaml`.

Antes de empezar

- Por defecto, el instalador de Trident despliega el operador en el `trident` namespace. Si el `trident` namespace no existe, créalo usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para desplegar el operador en un espacio de nombres distinto del `trident` namespace, actualiza `serviceaccount.yaml`, `clusterrolebinding.yaml` y `operator.yaml` y genera tu archivo `bundle` usando el `kustomization.yaml`.

- a. Crea el `kustomization.yaml` usando el siguiente comando donde `<bundle.yaml>` es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes.

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- b. Compila el paquete usando el siguiente comando donde `<bundle.yaml>` es `bundle_pre_1_25.yaml` o `bundle_post_1_25.yaml` según tu versión de Kubernetes.

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

Pasos

1. Crea los recursos y despliega el operador:

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

2. Verifica que se hayan creado el operador, el deployment y los replicaset.

```
kubectl get all -n <operator-namespace>
```



Solo debe haber **una instancia** del operador en un clúster de Kubernetes. No crees varias implementaciones del operador Trident.

Paso 5: actualiza la ubicación del registro de la imagen en la `TridentOrchestrator`

Tu "[Imágenes de Trident y CSI](#)" puede estar en un registro o en registros diferentes, pero todas las imágenes CSI deben estar en el mismo registro. Actualiza `deploy/crds/tridentorchestrator_cr.yaml` para agregar las especificaciones de ubicación adicionales según la configuración de tu registro.

Imágenes en un registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Imágenes en diferentes registros

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Paso 6: crea el `TridentOrchestrator` e instala Trident

Ahora puedes crear el `TridentOrchestrator` e instalar Trident. Opcionalmente, puedes ampliar aún más "[personaliza tu instalación Trident](#)" usando los atributos en la `TridentOrchestrator` spec. El siguiente ejemplo muestra una instalación donde las imágenes de Trident y CSI están ubicadas en diferentes registros.

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image: <your-registry>/trident-autosupport:26.02
  Debug:              true
  Image Registry:    <your-registry>
  Namespace:         trident
  Trident Image:     <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:    <your-registry>
    k8sTimeout:        30
    Kubelet Dir:       /var/lib/kubelet
    Log Format:         text
    Probe Port:        17546
    Silence Autosupport: false
    Trident Image:     <your-registry>/trident:26.02.0
  Message:             Trident installed
  Namespace:           trident
  Status:              Installed
  Version:             v26.02.0
Events:
  Type Reason Age From Message -----Normal
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

Verifica la instalación

Hay varias formas de verificar tu instalación.

Usando `TridentOrchestrator status`

El estado de `TridentOrchestrator` indica si la instalación fue exitosa y muestra la versión de Trident instalada. Durante la instalación, el estado de `TridentOrchestrator` cambia de `Installing` a `Installed`. Si observas el estado de `Failed` y el operador no puede recuperarse por sí solo, ["revisa los registros"](#).

Estado	Descripción
Instalando	El operador está instalando Trident usando este <code>TridentOrchestrator CR</code> .
Instalado	Trident se ha instalado correctamente.
Desinstalar	El operador está desinstalando Trident porque <code>spec.uninstall=true</code> .
Desinstalado	Trident está desinstalado.
Con errores	El operador no pudo instalar, parchear, actualizar o desinstalar Trident; el operador intentará automáticamente recuperarse de este estado. Si este estado persiste necesitarás hacer una solución de problemas.
Actualizando	El operador está actualizando una instalación existente.
Error	El <code>TridentOrchestrator</code> no se usa. Ya existe otro.

Uso del estado de creación de pods

Puedes confirmar si la instalación de Trident se completó revisando el estado de los pods creados:

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
AGE			
trident-controller-7d466bf5c7-v4cpw	6/6	Running	0
1m			
trident-node-linux-mr6zc	2/2	Running	0
1m			
trident-node-linux-xrp7w	2/2	Running	0
1m			
trident-node-linux-zh2jt	2/2	Running	0
1m			
trident-operator-766f7b8658-ldzsv	1/1	Running	0
3m			

Usando `tridentctl`

Puedes usar `tridentctl` para comprobar la versión de Trident instalada.

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.02.0        | 26.02.0        |
+-----+-----+
```

Implementa el operador de Trident usando Helm (modo estándar)

Puedes desplegar el operador Trident e instalar Trident usando Helm. Este proceso aplica a instalaciones donde las imágenes de contenedor requeridas por Trident no están almacenadas en un registro privado. Si tienes un registro de imágenes privado, usa el ["proceso para el despliegue fuera de línea"](#).

Información crítica sobre Trident 25.10

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.35 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Implementa el operador Trident e instala Trident usando Helm

Usando el ["Helm Chart"](#) de Trident, puedes desplegar el operador Trident e instalar Trident en un solo paso.

Revisa ["la visión general de la instalación"](#) para asegurarte de que cumpliste los requisitos previos de instalación y seleccionaste la opción de instalación correcta para tu entorno.

Antes de empezar

Además de ["requisitos previos de despliegue"](#) tú necesitas ["Helm versión 3"](#).

Pasos

1. Agrega el repositorio Trident Helm:

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

2. Usa `helm install` y especifica un nombre para tu despliegue como en el siguiente ejemplo donde `100..0` es la versión de Trident que estás instalando.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0  
--create-namespace --namespace <trident-namespace>
```



Si ya creaste un espacio de nombres para Trident, el parámetro `--create-namespace` no creará un espacio de nombres adicional.

Puedes usar `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, chart, estado, versión de la app y número de revisión.

Pasa los datos de configuración durante la instalación

Hay dos formas de pasar datos de configuración durante la instalación:

Opción	Descripción
<code>--values (o -f)</code>	Especifica un archivo YAML con anulaciones. Esto se puede especificar varias veces y el archivo más a la derecha tendrá prioridad.
<code>--set</code>	Especifica las modificaciones en la línea de comandos.


Por ejemplo, para cambiar el valor por defecto de `debug`, ejecuta el siguiente comando donde `100.2602.0` es la versión de Trident que estás instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0  
--create-namespace --namespace trident --set tridentDebug=true
```

Opciones de configuración

Esta tabla y el archivo `values.yaml`, que es parte del Helm chart, proporcionan la lista de claves y sus valores predeterminados.


Opción	Descripción	Predeterminado
<code>nodeSelector</code>	Etiquetas de nodo para la asignación de pod	
<code>podAnnotations</code>	Anotaciones del pod	


Opción	Descripción	Predeterminado
deploymentAnnotations	Anotaciones de deployment	
tolerations	Tolerancias para la asignación de pods	
affinity	Afinidad para la asignación de pods	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>No elimines la afinidad predeterminada del archivo values.yaml. Cuando quieras proporcionar una afinidad personalizada, amplía la afinidad predeterminada.</p> </div>
tridentControllerPluginNodeSelector	Selectores de nodos adicionales para pods. Consulta Entender los pods de controlador y los pods de nodo para más detalles.	
tridentControllerPluginTolerations	Anula las tolerancias de Kubernetes para los pods. Consulta Entender los pods de controlador y los pods de nodo para más detalles.	
tridentNodePluginNodeSelector	Selectores de nodos adicionales para pods. Consulta Entender los pods de controlador y los pods de nodo para más detalles.	

Opción	Descripción	Predeterminado
tridentNodePluginTolerations	Anula las tolerancias de Kubernetes para los pods. Consulta Entender los pods de controlador y los pods de nodo para más detalles.	
imageRegistry	Identifica el registro para el trident-operator, trident y otras imágenes. Déjalo vacío para aceptar el valor predeterminado. IMPORTANTE: cuando instales Trident en un repositorio privado, si usas el switch imageRegistry para especificar la ubicación del repositorio, no uses /netapp/ en la ruta del repositorio.	""
imagePullPolicy	Establece la política de extracción de imágenes para el trident-operator.	IfNotPresent
imagePullSecrets	Establece los secretos de extracción de imágenes para las trident-operator, trident y otras imágenes.	
kubeletDir	Permite anular la ubicación del host del estado interno de kubelet.	"/var/lib/kubelet"
operatorLogLevel	Permite establecer el nivel de registro del operador Trident en: trace, debug, info, warn, error o fatal.	"info"
operatorDebug	Permite establecer el nivel de registro del operador Trident en modo debug.	true
operatorImage	Permite la anulación completa de la imagen para trident-operator.	""
operatorImageTag	Permite anular la etiqueta de la trident-operator imagen.	""
tridentIPv6	Permite habilitar Trident para trabajar en clústeres IPv6.	false
tridentK8sTimeout	Anula el tiempo de espera predeterminado de 30 segundos para la mayoría de las operaciones de la API de Kubernetes (si no es cero, en segundos).	0

Opción	Descripción	Predeterminado
tridentHttpRequestTimeout	Anula el tiempo de espera predeterminado de 90 segundos para las solicitudes HTTP, con 0s siendo una duración infinita para el tiempo de espera. No se permiten valores negativos.	"90s"
tridentSilenceAutoSupport	Permite deshabilitar los informes periódicos de AutoSupport de Trident.	false
tridentAutoSupportImageTag	Permite sobrescribir la etiqueta de la imagen para el contenedor Trident AutoSupport.	<version>
tridentAutoSupportProxy	Permite que el contenedor Trident AutoSupport se comuniquen con la central a través de un proxy HTTP.	""
tridentLogFormat	Establece el formato de registro de Trident (text o json).	"text"
tridentDisableAuditLog	Desactiva el auditor de registros de Trident.	true
tridentLogLevel	Permite establecer el nivel de registro de Trident en: trace, debug, info, warn, error o fatal.	"info"
tridentDebug	Permite establecer el nivel de registro de Trident en debug. Puedes automatizar el proceso de desinstalación forzosa mediante la integración con el operador node health check (NHC). Para más información, consulta "Automatizando la conmutación por error de aplicaciones con estado con Trident" .	false
tridentLogWorkflows	Permite habilitar flujos de trabajo específicos de Trident para el registro de trazas o la supresión de logs.	""
tridentLogLayers	Permite habilitar capas específicas de Trident para el registro de trazas o la supresión de logs.	""
tridentImage	Permite la anulación completa de la imagen para Trident.	""
tridentImageTag	Permite anular la etiqueta de la imagen para Trident.	""

Opción	Descripción	Predeterminado
tridentProbePort	Permite anular el puerto predeterminado usado para las sondas de liveness/readiness de Kubernetes.	""
windows	Permite que Trident se instale en el nodo trabajador de Windows.	false
enableForceDetach	Permite activar la función de desconexión forzosa.	false
excludePodSecurityPolicy	Excluye la política de seguridad del pod de operador de la creación.	false
cloudProvider	Establécelo en "Azure" cuando uses identidades gestionadas o una identidad en la nube en un clúster AKS. Establécelo en "AWS" cuando uses una identidad en la nube en un clúster EKS.	""
cloudIdentity	Configura la identidad de carga de trabajo ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx") cuando uses cloud identity en un clúster AKS. Configura el rol de AWS IAM ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role") cuando uses cloud identity en un clúster EKS.	""
iscsiSelfHealingInterval	El intervalo en el que se invoca la reparación automática de iSCSI.	5m0s
iscsiSelfHealingWaitTime	La duración después de la cual la reparación automática iSCSI inicia un intento de resolver una sesión obsoleta realizando un cierre de sesión y un inicio de sesión posterior.	7m0s
nodePrep	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. Nota: a partir de OpenShift 4.19, la versión mínima de Trident admitida para esta función es 25.06.1.	

Opción	Descripción	Predeterminado
enableConcurrency	<p>Permite operaciones simultáneas del controlador Trident para mejorar el rendimiento.</p> <p> Tech Preview: Esta función es experimental y actualmente admite flujos de trabajo paralelos limitados con los controladores ONTAP-NAS (NFS solo) y ONTAP-SAN (NVMe para unified ONTAP 9), además de la tech preview existente para el controlador ONTAP-SAN (protocolos iSCSI y FCP en unified ONTAP 9).</p>	false
k8sAPIQPS	<p>El límite de consultas por segundo (QPS) que usa el controlador mientras se comunica con el servidor de la API de Kubernetes. El valor de ráfaga se establece automáticamente según el valor de QPS.</p>	100; opcional

Opción	Descripción	Predeterminado
resources	<p>Establece los límites y las solicitudes de recursos de Kubernetes para los pods del controlador, nodo y operador de Trident. Puedes configurar la CPU y la memoria de cada contenedor y sidecar para gestionar la asignación de recursos en Kubernetes.</p> <p>Para obtener más información sobre cómo configurar las solicitudes y los límites de recursos, consulta "Gestión de recursos para pods y contenedores".</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <ul style="list-style-type: none"> NO cambies los nombres de ningún contenedor ni de ningún campo. NO cambies la sangría: la sangría de YAML es fundamental para que se analice correctamente. </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node: linux: trident-main:</pre>

Opción	Descripción	Predeterminado
httpsMetrics	Habilita HTTPS para el endpoint de métricas de Prometheus.	false
hostNetwork	Habilita la conexión en red del host para el controlador Trident. Esto es útil cuando quieres separar el tráfico de frontend y backend en una red multi-home.	false

Entender los pods de controlador y los pods de nodo

Trident se ejecuta como un pod controlador único, más un pod de nodo en cada nodo trabajador del clúster. El pod de nodo debe estar ejecutándose en cualquier host donde quieras montar potencialmente un volumen Trident.

Kubernetes "selectores de nodos" y "tolerancias y taints" se usan para restringir que un pod se ejecute en un nodo específico o preferido. Usando el ControllerPlugin y NodePlugin, puedes especificar restricciones y anulaciones.

- Los sidecars aparecen debajo de cada contenedor principal.
- El complemento del controlador se encarga del aprovisionamiento y la gestión de volúmenes, como las instantáneas y el cambio de tamaño.
- El plugin de nodo se encarga de conectar el almacenamiento al nodo.

Implementa el operador de Trident usando Helm (modo sin conexión)

Puedes desplegar el operador Trident e instalar Trident usando Helm. Este proceso aplica a instalaciones donde los volúmenes de contenedor requeridos por Trident se almacenan en un registro privado. Si no tienes un registro de imágenes privado, usa el "proceso para la implementación estándar".

Información crítica sobre Trident 26.02

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

```
requests:
  cpu: 1m
  memory: 10Mi
limits:
  cpu:
  memory:
windows:
  trident-agent:
requests:
  cpu: 6m
  memory: 40Mi
limits:
  cpu:
  memory:
node-driver-registry:
requests:
  cpu: 6m
  memory: 40Mi
limits:
  cpu:
  memory:
liveness-probe:
requests:
  cpu: 2m
  memory: 40Mi
```

- Kubernetes 1.35 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Implementa el operador Trident e instala Trident usando Helm

Usando el ["Helm Chart"](#) de Trident, puedes desplegar el operador Trident e instalar Trident en un solo paso.

Revisa ["la visión general de la instalación"](#) para asegurarte de que cumpliste los requisitos previos de instalación y seleccionaste la opción de instalación correcta para tu entorno.

Antes de empezar

Además de ["requisitos previos de despliegue"](#) tú necesitas ["Helm versión 3"](#).



Cuando instales Trident en un repositorio privado, si estás usando el modificador `imageRegistry` para especificar la ubicación del repositorio, no uses `/netapp/` en la ruta del repositorio.

Pasos

1. Agrega el repositorio Trident Helm:

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

2. Usa `helm install` y especifica un nombre para tu deployment y la ubicación del image registry. Tu ["Imágenes de Trident y CSI"](#) puede estar en un registro o en registros diferentes, pero todas las imágenes CSI deben estar en el mismo registro. En los ejemplos, `100.2602.0` es la versión de Trident que estás instalando.

Imágenes en un registro

```
helm install <name> netapp-trident/trident-operator --version  
100.2602.0 --set imageRegistry=<your-registry> --create-namespace  
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Imágenes en diferentes registros

```
helm install <name> netapp-trident/trident-operator --version  
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage  
=<your-registry>/trident-operator:26.02.0 --set  
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02  
--set tridentImage=<your-registry>/trident:26.02.0 --create  
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



Si ya creaste un espacio de nombres para Trident, el parámetro `--create-namespace` no creará un espacio de nombres adicional.

Puedes usar `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, chart, estado, versión de la app y número de revisión.

Pasa los datos de configuración durante la instalación

Hay dos formas de pasar datos de configuración durante la instalación:

Opción	Descripción
<code>--values (o -f)</code>	Especifica un archivo YAML con anulaciones. Esto se puede especificar varias veces y el archivo más a la derecha tendrá prioridad.
<code>--set</code>	Especifica las modificaciones en la línea de comandos.

Por ejemplo, para cambiar el valor por defecto de `debug`, ejecuta el siguiente comando donde `100.2602.0` es la versión de Trident que estás instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Para añadir el valor `nodePrep`, ejecuta el siguiente comando:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```


Opciones de configuración

Esta tabla y el archivo `values.yaml`, que es parte del Helm chart, proporcionan la lista de claves y sus valores predeterminados.




No elimines la afinidad predeterminada del archivo `values.yaml`. Cuando quieras proporcionar una afinidad personalizada, amplía la afinidad predeterminada.


Opción	Descripción	Predeterminado
<code>nodeSelector</code>	Etiquetas de nodo para la asignación de pod	
<code>podAnnotations</code>	Anotaciones del pod	
<code>deploymentAnnotations</code>	Anotaciones de deployment	
<code>tolerations</code>	Tolerancias para la asignación de pods	

Opción	Descripción	Predeterminado
affinity	Afinidad para la asignación de pods	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>No elimines la afinidad predeterminada del archivo values.yaml. Cuando quieras proporcionar una afinidad personalizada, amplía la afinidad predeterminada.</p> </div>
tridentControllerPluginNodeSelector	Selectores de nodos adicionales para pods. Consulta "Entender los pods de controlador y los pods de nodo" para más detalles.	
tridentControllerPluginTolerations	Anula las tolerancias de Kubernetes para los pods. Consulta "Entender los pods de controlador y los pods de nodo" para más detalles.	

Opción	Descripción	Predeterminado
<code>tridentNodePluginNodeSelector</code>	Selectores de nodos adicionales para pods. Consulta "Entender los pods de controlador y los pods de nodo" para más detalles.	
<code>tridentNodePluginTolerations</code>	Anula las tolerancias de Kubernetes para los pods. Consulta "Entender los pods de controlador y los pods de nodo" para más detalles.	
<code>imageRegistry</code>	Identifica el registro para el <code>trident-operator</code> , <code>trident</code> y otras imágenes. Déjalo vacío para aceptar el valor predeterminado. IMPORTANTE: cuando instales Trident en un repositorio privado, si usas el switch <code>imageRegistry</code> para especificar la ubicación del repositorio, no uses <code>/netapp/</code> en la ruta del repositorio.	""
<code>imagePullPolicy</code>	Establece la política de extracción de imágenes para el <code>trident-operator</code> .	IfNotPresent
<code>imagePullSecrets</code>	Establece los secretos de extracción de imágenes para las <code>trident-operator</code> , <code>trident</code> y otras imágenes.	
<code>kubeletDir</code>	Permite anular la ubicación del host del estado interno de kubelet.	<code>"/var/lib/kubelet"</code>
<code>operatorLogLevel</code>	Permite establecer el nivel de registro del operador Trident en: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> o <code>fatal</code> .	<code>"info"</code>
<code>operatorDebug</code>	Permite establecer el nivel de registro del operador Trident en modo debug.	<code>true</code>
<code>operatorImage</code>	Permite la anulación completa de la imagen para <code>trident-operator</code> .	""
<code>operatorImageTag</code>	Permite anular la etiqueta de la <code>trident-operator</code> imagen.	""
<code>tridentIPv6</code>	Permite habilitar Trident para trabajar en clústeres IPv6.	<code>false</code>

Opción	Descripción	Predeterminado
<code>tridentK8sTimeout</code>	<p>Anula el tiempo de espera predeterminado de 180 segundos para la mayoría de las operaciones de la API de Kubernetes (si no es cero, en segundos).</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>El <code>tridentK8sTimeout</code> parámetro solo es aplicable para la instalación de Trident.</p> </div>	180
<code>tridentHttpRequestTimeout</code>	Anula el tiempo de espera predeterminado de 90 segundos para las solicitudes HTTP, con 0s siendo una duración infinita para el tiempo de espera. No se permiten valores negativos.	"90s"
<code>tridentSilenceAutosupport</code>	Permite deshabilitar los informes periódicos de AutoSupport de Trident.	false
<code>tridentAutosupportImageTag</code>	Permite sobrescribir la etiqueta de la imagen para el contenedor Trident AutoSupport.	<version>
<code>tridentAutosupportProxy</code>	Permite que el contenedor Trident AutoSupport se comuniquen con la central a través de un proxy HTTP.	""
<code>tridentLogFormat</code>	Establece el formato de registro de Trident (<code>text</code> o <code>json</code>).	"text"
<code>tridentDisableAuditLog</code>	Desactiva el auditor de registros de Trident.	true
<code>tridentLogLevel</code>	Permite establecer el nivel de registro de Trident en: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> o <code>fatal</code> .	"info"
<code>tridentDebug</code>	Permite establecer el nivel de registro de Trident en <code>debug</code> .	false
<code>tridentLogWorkflows</code>	Permite habilitar flujos de trabajo específicos de Trident para el registro de trazas o la supresión de logs.	""
<code>tridentLogLayers</code>	Permite habilitar capas específicas de Trident para el registro de trazas o la supresión de logs.	""

Opción	Descripción	Predeterminado
tridentImage	Permite la anulación completa de la imagen para Trident.	""
tridentImageTag	Permite anular la etiqueta de la imagen para Trident.	""
tridentProbePort	Permite anular el puerto predeterminado usado para las sondas de liveness/readiness de Kubernetes.	""
windows	Permite que Trident se instale en el nodo trabajador de Windows.	false
enableForceDetach	Permite activar la función de desconexión forzosa. Puedes automatizar el proceso de desinstalación forzosa mediante la integración con el operador node health check (NHC). Para más información, consulta "Automatizando la conmutación por error de aplicaciones con estado con Trident" .	false
excludePodSecurityPolicy	Excluye la política de seguridad del pod de operador de la creación.	false
nodePrep	<p>Permite que Trident prepare los nodos del clúster de Kubernetes para gestionar volúmenes usando el protocolo de almacenamiento de datos especificado. Actualmente, <code>iscsi</code> es el único valor admitido.</p> <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-top: 10px;">  <p>A partir de OpenShift 4.19, la versión mínima de Trident compatible con esta función es 25.06.1.</p> </div>	

Opción	Descripción	Predeterminado
resources	<p>Establece los límites y las solicitudes de recursos de Kubernetes para los pods del controlador, nodo y operador de Trident. Puedes configurar la CPU y la memoria de cada contenedor y sidecar para gestionar la asignación de recursos en Kubernetes.</p> <p>Para obtener más información sobre cómo configurar las solicitudes y los límites de recursos, consulta "Gestión de recursos para pods y contenedores".</p> <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;">  <ul style="list-style-type: none"> • NO cambies los nombres de ningún contenedor ni de ningún campo. • NO cambies la sangría: la sangría de YAML es fundamental para que se analice correctamente. </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident- autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node: linux:</pre>

Personaliza la instalación del operador Trident

El operador Trident te permite personalizar la instalación de Trident usando los atributos en la `TridentOrchestrator` spec. Si quieres personalizar la instalación más allá de lo que permiten los `TridentOrchestrator` arguments, considera usar `tridentctl` para generar manifiestos YAML personalizados y modificarlos como necesites.

Entender los pods de controlador y los pods de nodo

Trident se ejecuta como un único pod de controlador y un pod de nodo en cada nodo de trabajo del clúster. El pod de nodo debe estar ejecutándose en cualquier host donde quieras montar potencialmente un volumen Trident.

Kubernetes "selectores de nodos" y "tolerancias y taints" se usan para restringir que un pod se ejecute en un nodo específico o preferido. Usando ControllerPlugin y NodePlugin, puedes especificar restricciones y anulaciones.

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

Opciones de configuración



`spec.namespace` se especifica en `TridentOrchestrator` para indicar el namespace donde está instalado Trident. Este parámetro **no se puede actualizar después de instalar Trident**. Si intentas hacerlo, el estado de `TridentOrchestrator` cambia a `Failed`. Trident no está pensado para migrarse entre namespaces. Los valores que están aplicados ahora.


Esta tabla detalla `TridentOrchestrator` los atributos.



Parámetro	Descripción	Predeterminado
<code>namespace</code>	Espacio de nombres para instalar Trident	"default"
<code>debug</code>	Habilita la depuración para Trident	false
<code>enableForceDetach</code>	<code>ontap-san</code> , <code>ontap-san-economy</code> , <code>ontap-nas</code> y <code>ontap-nas-economy</code> solamente. Funciona con Kubernetes Non-Graceful Node Shutdown (NGNS) para que los administradores de clúster tengan la capacidad de migrar de forma segura cargas de trabajo con volúmenes montados a nuevos nodos si un nodo deja de estar saludable. Para más información, consulta "Automatizando la conmutación por error de aplicaciones con estado con Trident" .	false
<code>windows</code>	La configuración de <code>true</code> permite la instalación en nodos worker de Windows.	false


```

trident-main:
  requests:
    cpu: 10m
    memory: 60Mi
  limits:
    cpu:
    memory:
  registrar:
  requests:
    cpu: 10m
    memory: 10Mi
  limits:
    cpu:
    memory:
  windows:
    trident-main:
      requests:
        cpu: 6m
        memory: 40Mi
      limits:
        cpu:
        memory:
  operator:
  requests:
    cpu: 10m
    memory: 40Mi
  limits:

```

Parámetro	Descripción	Predeterminado
cloudProvider	Establécelo en "Azure" cuando uses identidades gestionadas o una identidad en la nube en un clúster AKS. Configúralo en "AWS" cuando uses una identidad en la nube en un clúster de EKS. Configúralo en "GCP" cuando uses una identidad en la nube en un clúster de GKE.	""
cloudIdentity	Configura la identidad de carga de trabajo ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") cuando uses cloud identity en un clúster AKS. Configura el rol de AWS IAM ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role") cuando uses cloud identity en un clúster EKS. Configura la cloud identity ("iam.gke.io/gcp-service-account: xxx@mygcpproject.iam.gserviceaccount.com") cuando uses cloud identity en un clúster de GKE.	""
IPv6	Instala Trident sobre IPv6	false
k8sTimeout	Tiempo de espera para las operaciones de Kubernetes.  El k8sTimeout parámetro solo es aplicable para la instalación de Trident.	180sec
silenceAutosupport	No envíes paquetes de autosupport a NetApp automáticamente	false
autosupportImage	La imagen del contenedor para Autosupport Telemetry	"netapp/trident-autosupport10"
autosupportProxy	La dirección/puerto de un proxy para enviar Autosupport Telemetry	"http://proxy.example.com:8888"
uninstall	Una bandera usada para desinstalar Trident	false
logFormat	Formato de registro de Trident que se usará [text,json]	"text"
tridentImage	Imagen de Trident para instalar	"netapp/trident:26.02"
imageRegistry	Ruta al registro interno, de formato <registry FQDN>[:port][subpath]	"registry.k8s.io"
kubeletDir	Ruta al directorio de kubelet en el host	"/var/lib/kubelet"
wipeout	Una lista de recursos para eliminar y así realizar una eliminación completa de Trident	
imagePullSecrets	Secretos para extraer imágenes de un registro interno	

Parámetro	Descripción	Predeterminado
imagePullPolicy	Establece la política de extracción de imágenes para el operador Trident. Los valores válidos son: Always para extraer siempre la imagen. IfNotPresent para extraer la imagen solo si no existe en el nodo. Never para nunca extraer la imagen.	IfNotPresent
controllerPluginNodeSelector	Selectores de nodos adicionales para pods. Sigue el mismo formato que <code>pod.spec.nodeSelector</code> .	Sin valor predeterminado; opcional
controllerPluginTolerations	Anula las tolerancias de Kubernetes para los pods. Sigue el mismo formato que <code>pod.spec.Tolerations</code> .	Sin valor predeterminado; opcional
nodePluginNodeSelector	Selectores de nodos adicionales para pods. Sigue el mismo formato que <code>pod.spec.nodeSelector</code> .	Sin valor predeterminado; opcional
nodePluginTolerations	Anula las tolerancias de Kubernetes para los pods. Sigue el mismo formato que <code>pod.spec.Tolerations</code> .	Sin valor predeterminado; opcional
nodePrep	Permite que Trident prepare los nodos del clúster de Kubernetes para gestionar volúmenes usando el protocolo de almacenamiento de datos especificado. Actualmente, <code>iscsi</code> es el único valor admitido. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  A partir de OpenShift 4.19, la versión mínima de Trident compatible con esta función es 25.06.1. </div>	
k8sAPIQPS	El límite de consultas por segundo (QPS) que usa el controlador mientras se comunica con el servidor de la API de Kubernetes. El valor de ráfaga se establece automáticamente según el valor de QPS.	100; opcional
enableConcurrency	Permite operaciones simultáneas del controlador Trident para mejorar el rendimiento. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Tech Preview: Esta función es experimental y actualmente admite flujos de trabajo paralelos limitados con los controladores ONTAP-NAS (NFS solo) y ONTAP-SAN (NVMe para unified ONTAP 9), además de la tech preview existente para el controlador ONTAP-SAN (protocolos iSCSI y FCP en unified ONTAP 9). </div>	false

Parámetro	Descripción	Predeterminado
resources	<p data-bbox="475 157 1130 325">Establece los límites y las solicitudes de recursos de Kubernetes para el controlador Trident y los pods de nodo. Puedes configurar la CPU y la memoria de cada contenedor y sidecar para gestionar la asignación de recursos en Kubernetes.</p> <p data-bbox="475 359 1138 457">Para obtener más información sobre cómo configurar las solicitudes y los límites de recursos, consulta "Gestión de recursos para pods y contenedores".</p> <div data-bbox="505 493 1112 1312" style="border-left: 1px solid #ccc; padding-left: 10px;"> <ul style="list-style-type: none"> <li data-bbox="505 583 565 646"> <li data-bbox="646 506 1101 724"> <ul style="list-style-type: none"> • NO cambies los nombres de ningún contenedor ni de ningún campo. • NO cambies la sangría: la sangría de YAML es fundamental para que se analice correctamente. <li data-bbox="505 1014 565 1077"> • No se aplican límites por defecto: solo las solicitudes tienen valores por defecto y se aplican automáticamente si no se especifican. • Los nombres de los contenedores figuran tal como aparecen en las especificaciones del pod. • Los sidecars aparecen debajo de cada contenedor principal. • Revisa el campo <code>status.CurrentInstallationParams</code> del TORC para ver los valores que están aplicados ahora. </div>	<pre data-bbox="1182 191 1463 2074">resources: controller: trident- main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi- provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi- attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi- resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi- snapshotter: requests: cpu: 2m memory: 20Mi limits:</pre>

Parámetro	Descripción	Predeterminado
httpsMetrics	Habilita HTTPS para el endpoint de métricas de Prometheus.	false
hostNetwork	Habilita la conexión en red del host para el controlador Trident. Esto es útil cuando quieres separar el tráfico de frontend y backend en una red multi-home.	false



Para más información sobre cómo formatear los parámetros del pod, consulta ["Asignar pods a nodos"](#).

```

30Mi
limits:
  cpu:
  memory:
node:
linux:
  trident-
main:

```

Configuraciones de ejemplo

Puedes usar los atributos en [Opciones de configuración](#) cuando defines `TridentOrchestrator` para personalizar tu instalación.

Configuración básica personalizada

Este ejemplo, creado después de ejecutar el `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` comando, representa una instalación personalizada básica:

```

apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret

```

```

lm
memory: 10Mi
  limits:
    cpu:
memory:
  windows:
    trident-
main:
requests:
  cpu:
6m
memory: 40Mi
  limits:

```

Selectores de nodos

Este ejemplo instala Trident con selectores de nodos.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

```
memory:
liveness-
```

Nodos trabajadores de Windows

Este ejemplo, creado después de ejecutar el `cat deploy/crds/tridentorchestrator_cr.yaml` comando, instala Trident en un nodo trabajador Windows.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Identities gestionadas en un clúster de AKS

Este ejemplo instala Trident para habilitar identidades gestionadas en un clúster de AKS.

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

Cloud identity en un clúster AKS

Este ejemplo instala Trident para usarlo con una identidad en la nube en un clúster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxxx'
```

Cloud identity en un clúster EKS

Este ejemplo instala Trident para usarlo con una identidad en la nube en un clúster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Identidad en la nube para GKE

Este ejemplo instala Trident para usarlo con una identidad en la nube en un clúster GKE.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Configuración de solicitudes y límites de recursos de Kubernetes para el controlador Trident y los pods de nodos Linux Trident

Este ejemplo configura las solicitudes y los límites de recursos de Kubernetes para el controlador Trident y los pods de nodo Linux Trident.



Descargo de responsabilidad: Los valores de solicitud y límite proporcionados en este ejemplo son solo para fines de demostración. Ajusta estos valores según tu entorno y los requisitos de carga de trabajo.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
    # sidecars
    csi-provisioner:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:
        cpu: 100m
        memory: 64Mi
    csi-attacher:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:
        cpu: 100m
        memory: 64Mi
    csi-resizer:
      requests:
        cpu: 3m
```

```
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
        cpu: 200m
        memory: 256Mi
    # sidecars
    node-driver-registrar:
      requests:
        cpu: 1m
        memory: 10Mi
      limits:
        cpu: 50m
        memory: 32Mi
```

Configuración de solicitudes y límites de recursos de Kubernetes para el controlador Trident y los pods de nodo Trident Windows y Linux

Este ejemplo configura las solicitudes y los límites de recursos de Kubernetes para el controlador Trident y los pods de nodo Trident Windows y Linux.



Descargo de responsabilidad: Los valores de solicitud y límite proporcionados en este ejemplo son solo para fines de demostración. Ajusta estos valores según tu entorno y los requisitos de carga de trabajo.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
    csi-provisioner:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:
        cpu: 100m
        memory: 64Mi
    csi-attacher:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:
        cpu: 100m
        memory: 64Mi
    csi-resizer:
      requests:
```

```
    cpu: 3m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
        cpu: 200m
        memory: 256Mi
    # sidecars
    node-driver-registrar:
      requests:
        cpu: 1m
        memory: 10Mi
      limits:
        cpu: 50m
        memory: 32Mi
  windows:
    trident-main:
      requests:
        cpu: 6m
        memory: 40Mi
      limits:
        cpu: 200m
        memory: 128Mi
    # sidecars
    node-driver-registrar:
```

```
requests:
  cpu: 6m
  memory: 40Mi
limits:
  cpu: 100m
  memory: 128Mi
liveness-probe:
  requests:
    cpu: 2m
    memory: 40Mi
  limits:
    cpu: 50m
    memory: 64Mi
```

Instala usando tridentctl

Instala usando tridentctl

Puedes instalar Trident usando `tridentctl`. Este proceso aplica a instalaciones donde las imágenes de contenedor requeridas por Trident se almacenan en un registro privado o no. Para personalizar tu despliegue `tridentctl`, consulta "[Personaliza el despliegue de tridentctl](#)".

Información crítica sobre Trident10

Debes leer la siguiente información crítica sobre Trident.

Información crítica sobre Trident

- Kubernetes 1.27 ya es compatible con Trident. Actualiza Trident antes de actualizar Kubernetes.
- Trident impone estrictamente el uso de la configuración de multivía en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo `multipath.conf`.

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

Instala Trident usando `tridentctl`

Revisa "[la visión general de la instalación](#)" para asegurarte de que cumpliste los requisitos previos de instalación y seleccionaste la opción de instalación correcta para tu entorno.

Antes de empezar

Antes de comenzar la instalación, inicia sesión en el host Linux y verifica que está gestionando un servidor en funcionamiento, "[clúster de Kubernetes compatible](#)" y que tienes los privilegios necesarios.



Con OpenShift, usa `oc` en vez de `kubectl` en todos los ejemplos que siguen, y primero inicia sesión como **system:admin** ejecutando `oc login -u system:admin o oc login -u kube-admin`.

1. Verifica tu versión de Kubernetes:

```
kubectl version
```

2. Verifica los privilegios del administrador del clúster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifica que puedes lanzar un pod que use una imagen de Docker Hub y acceder a tu sistema de almacenamiento a través de la red de pods:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

Paso 1: descarga el paquete de instalación de Trident

El paquete instalador de Trident crea un pod Trident, configura los objetos CRD que se usan para mantener su estado e inicializa los sidecars CSI para hacer cosas como aprovisionar y adjuntar volúmenes a los hosts del clúster. Descarga y extrae la última versión del instalador de Trident desde "[la sección Assets en GitHub](#)". Actualiza `<trident-installer-XX.XX.X.tar.gz>` en el ejemplo con la versión de Trident que elijas.

```
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
```

Paso 2: instalar Trident

Instala Trident en el espacio de nombres que quieras ejecutando el comando `tridentctl install`. Puedes añadir argumentos adicionales para especificar la ubicación del registro de imágenes.

Modo estándar

```
./tridentctl install -n trident
```

Imágenes en un registro

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident  
-image <your-registry>/trident:26.02.0
```

Imágenes en diferentes registros

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident  
-image <your-registry>/trident:26.02.0
```

El estado de tu instalación debería verse más o menos así.

```
....  
INFO Starting Trident installation.                namespace=trident  
INFO Created service account.  
INFO Created cluster role.  
INFO Created cluster role binding.  
INFO Added finalizers to custom resource definitions.  
INFO Created Trident service.  
INFO Created Trident secret.  
INFO Created Trident deployment.  
INFO Created Trident daemonset.  
INFO Waiting for Trident pod to start.  
INFO Trident pod started.                          namespace=trident  
pod=trident-controller-679648bd45-cv2mx  
INFO Waiting for Trident REST interface.  
INFO Trident REST interface is up.                 version=26.10.0  
INFO Trident installation succeeded.  
....
```

Verifica la instalación

Puedes verificar tu instalación usando el estado de creación de pods o `tridentctl`.

Uso del estado de creación de pods

Puedes confirmar si la instalación de Trident se completó revisando el estado de los pods creados:

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
trident-controller-679648bd45-cv2mx	6/6	Running	0	5m29s
trident-node-linux-vgc8n	2/2	Running	0	5m29s



Si el instalador no se completa correctamente o `trident-controller-<generated id>` (`trident-csi-<generated id>` en versiones anteriores a 23.01) no tiene un estado **Running**, la plataforma no se instaló. Usa `-d` para ["activa el modo de depuración"](#) y solucionar el problema.

Usando `tridentctl`

Puedes usar `tridentctl` para comprobar la versión de Trident instalada.

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.02.0       | 26.02.0       |
+-----+-----+
```

Configuraciones de ejemplo

Los siguientes ejemplos proporcionan configuraciones de muestra para instalar Trident usando `tridentctl`.

Nodos Windows

Para permitir que Trident se ejecute en nodos Windows:

```
tridentctl install --windows -n trident
```

Forzar separación

Para más información, consulta ["Automatizando la conmutación por error de aplicaciones con estado con Trident"](#).

```
tridentctl install --enable-force-detach=true -n trident
```

Habilita operaciones simultáneas del controlador Trident

Para habilitar operaciones simultáneas del controlador Trident para mejorar el rendimiento, agrega la opción `--enable-concurrency` durante la instalación como se muestra en este ejemplo.



Tech Preview: Esta función es experimental y actualmente admite flujos de trabajo paralelos limitados con los controladores ONTAP-NAS (NFS solo) y ONTAP-SAN (NVMe para unified ONTAP 9), además de la tech preview existente para el controlador ONTAP-SAN (protocolos iSCSI y FCP en unified ONTAP 9).

```
tridentctl install --enable-concurrency -n trident
```

Personaliza la instalación de tridentctl

Puedes usar el instalador Trident para personalizar la instalación.

Conoce el instalador

El instalador de Trident te permite personalizar atributos. Por ejemplo, si copiaste la imagen de Trident a un repositorio privado, puedes especificar el nombre de la imagen usando `--trident-image`. Si copiaste la imagen de Trident y también las imágenes CSI sidecar necesarias a un repositorio privado, podría ser mejor especificar la ubicación de ese repositorio usando el `--image-registry` switch, que tiene la forma `<registry FQDN>[:port]`.



Cuando instales Trident en un repositorio privado, si estás usando el modificador `--image-registry` para especificar la ubicación del repositorio, no uses `/netapp/` en la ruta del repositorio. Por ejemplo: `./tridentctl install --image-registry <image-registry> -n <namespace>`

Si estás usando una distribución de Kubernetes, donde `kubelet` mantiene sus datos en una ruta distinta a la habitual `/var/lib/kubelet`, puedes especificar la ruta alternativa usando `--kubelet-dir`.

Si necesitas personalizar la instalación más allá de lo que permiten los argumentos del instalador, también puedes personalizar los archivos de implementación. Usar el `--generate-custom-yaml` parámetro crea los siguientes archivos YAML en el directorio del instalador `setup`:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`

- trident-resourcequota.yaml *

Después de generar estos archivos, puedes modificarlos según tus necesidades y luego usar `--use-custom-yaml` para instalar tu implementación personalizada.

```
./tridentctl install -n trident --use-custom-yaml
```

Instala usando el operador certificado de OpenShift

Instala Trident usando OpenShift OperatorHub

Si usas Red Hat OpenShift, puedes instalar NetApp Trident usando el operador certificado de Red Hat. Usa este procedimiento para instalar Trident desde la plataforma Red Hat OpenShift Container Platform.

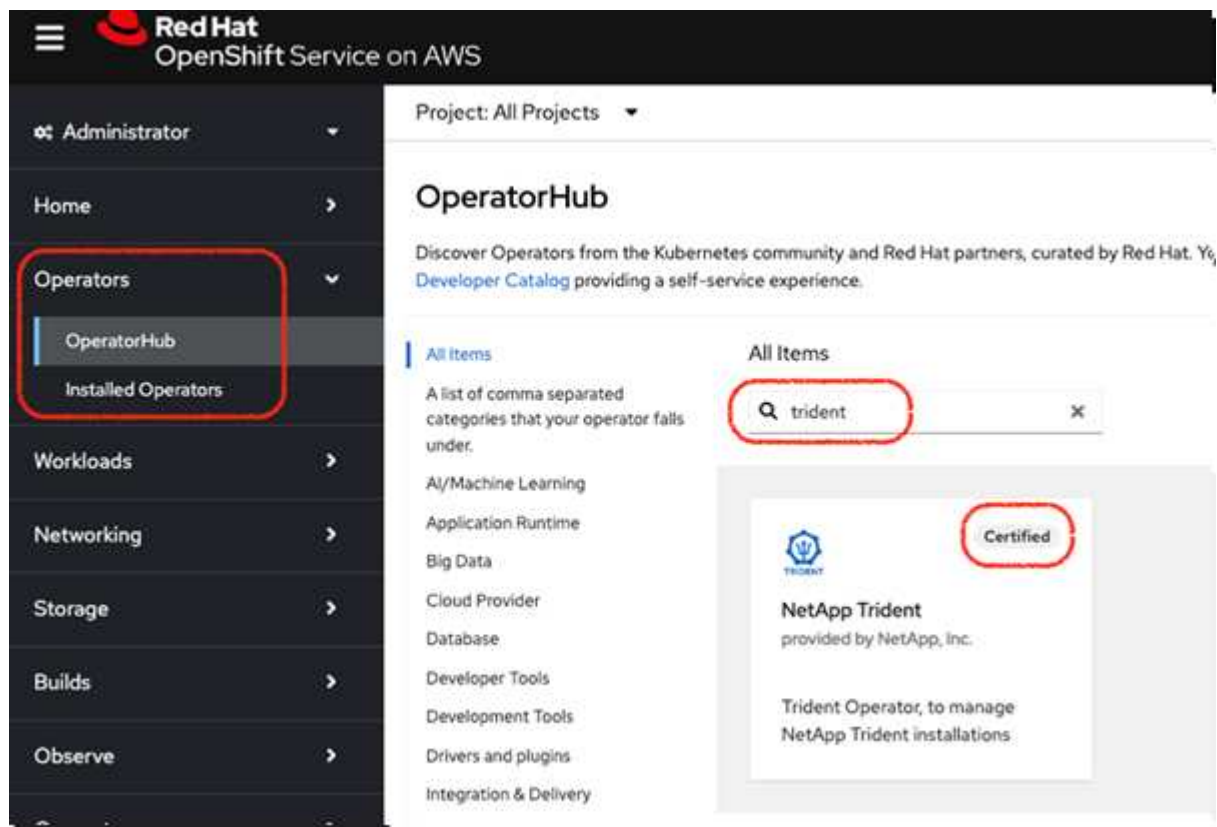
Antes de empezar

Antes de comenzar la instalación, ["prepara tu entorno para la instalación de Trident"](#).

Busca e instala el operador Trident

Pasos

1. Ve a OpenShift OperatorHub y busca NetApp Trident.



2. Haz clic en **NetApp Trident** para abrir la configuración de la instalación.
3. Selecciona las opciones necesarias y haz clic en **Instalar** para abrir la configuración del Operator.



NetApp Trident

25.2.1 provided by NetApp, Inc.



Install

Channel

stable

NetApp Trident is an open source storage provisioner and orchestrator maintained by NetApp. It enables you to create storage volumes for containerized applications managed by Docker and Kubernetes. For full release information, including patch release changes, see <https://docs.netapp.com/us-en/trident/trident-rn.html>.

Version

25.2.1

25.2.1

25.2.0

Source

Certified



Asegúrate de seleccionar la versión más reciente del Operator.

4. Mantén todos los parámetros como están y haz clic en **Instalar**.

OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

stable

Version *

25.2.1

Installation mode *

- All namespaces on the cluster (default)
Operator will be available in all Namespaces.
- A specific namespace on the cluster
This mode is not supported by this Operator

Installed Namespace *

openshift-operators

Update approval *

- Automatic
- Manual



NetApp Trident

provided by NetApp, Inc.

Provided APIs

TO Trident Orchestrator

Used to deploy NetApp Trident.

TC Trident Configurator

Automates AWS FSxN backend configuration

5. Haz clic en **Ver operador** para ver los detalles del operador.



Provided APIs

TO Trident Orchestrator

Used to deploy NetApp Trident.

[Create instance](#)

TC Trident Configurator

Automates AWS FSxN backend configuration

[Create instance](#)

6. Haz clic en **YAML view** y pega lo siguiente en el formulario:

```

apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false

```



La interfaz de usuario proporciona un ejemplo por defecto. Puedes editarlo directamente en lugar de copiar una configuración completa.

Opcional: activar la concurrencia



Para activar la concurrencia, agrega el siguiente campo a la especificación:

```
enableConcurrency: true
```



- Red Hat Enterprise Linux CoreOS (RHCOS) no tiene iSCSI habilitado ni configurado.
- Puedes añadir el parámetro `nodePrep` para configurar y habilitar tanto los servicios iSCSI como Multipath en todos los nodos worker de OpenShift.
- A partir de OpenShift 4.19, la versión mínima de Trident compatible con esta función es 25.06.1.

1. Haz clic en **Crear**; el orquestador Trident estará completamente instalado.

Name	Kind	Status	Labels	Last updated
trident	TridentOrchestrator	Status: Installed	No labels	Apr 6, 2025, 8:02 PM

Desinstalar Trident operator

Pasos

1. Selecciona el operador Trident de la lista de operadores instalados.
2. Selecciona si quieres eliminar todas las instancias de operand del operador.



Si no seleccionas la casilla **Eliminar todas las instancias de operandos de este operador**, Trident no se desinstalará.

3. Haz clic en **Desinstalar**.

Cambia al operador Trident certificado OpenShift

Puedes cambiar al operador Trident certificado de Red Hat OpenShift desde el operador de la comunidad, una instalación basada en Helm o un operador desplegado manualmente. El proceso para cada método implica desinstalar el operador existente y luego instalar el operador certificado usando el OperatorHub.

Antes de empezar

Antes de comenzar la instalación, ["prepara tu entorno para la instalación de Trident"](#).

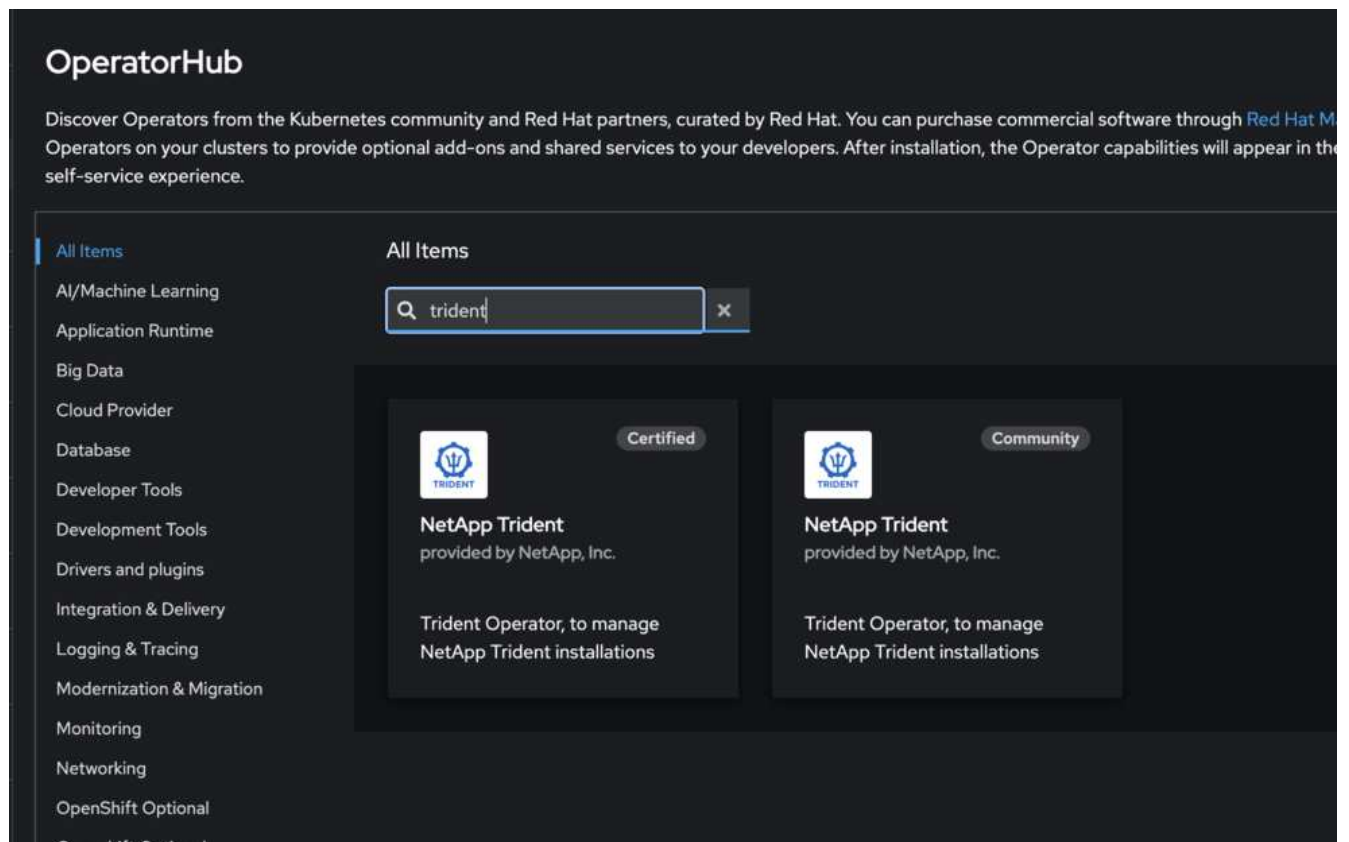


No elimines el `TridentOrchestrator` recurso personalizado (CR) durante el proceso de desinstalación. El `TridentOrchestrator` CR conserva tu configuración de backend y clase de almacenamiento, que se necesita después de instalar el operador certificado.

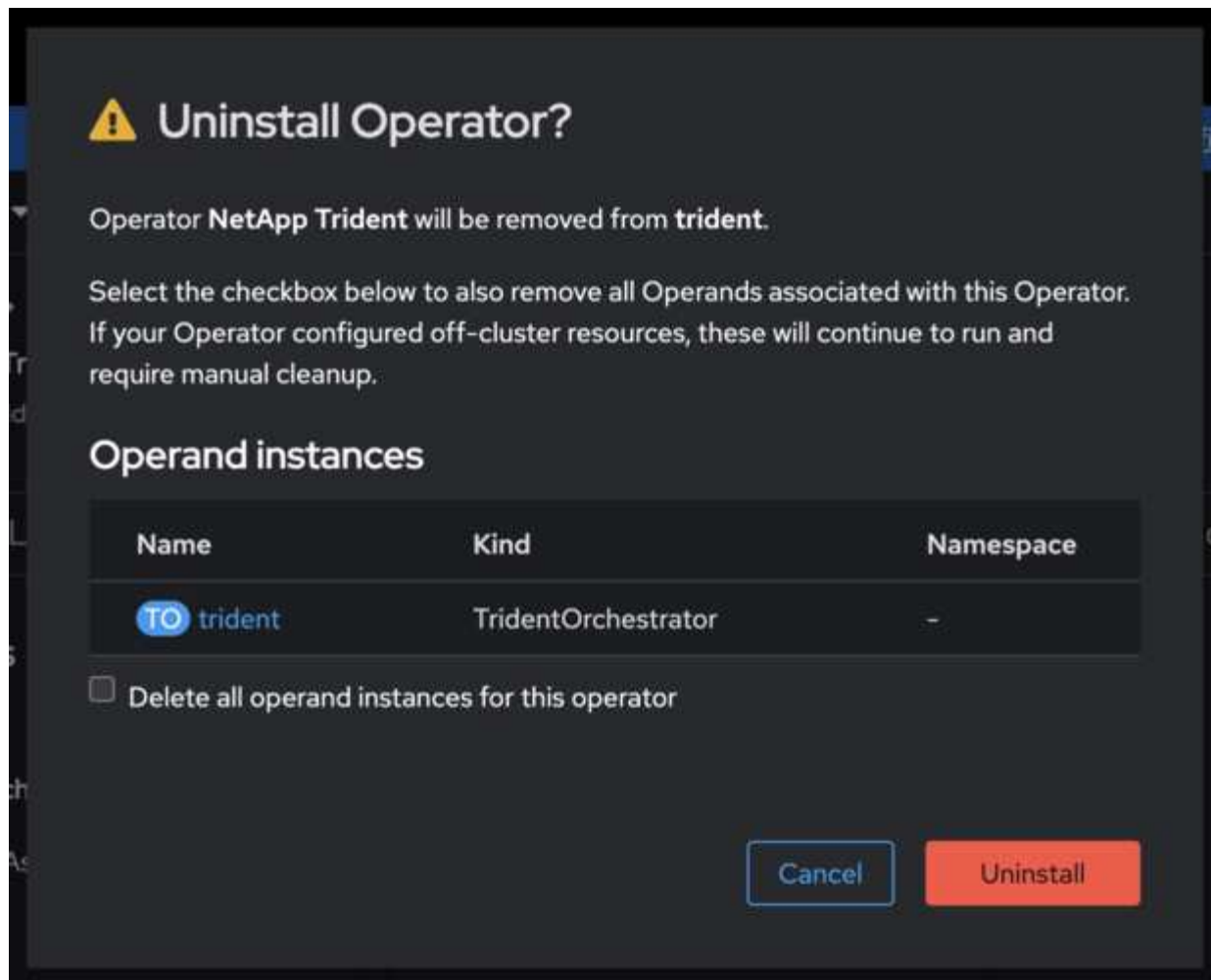
Cambia desde el operador comunitario

Pasos

1. Usa la consola de OpenShift para navegar hasta OperatorHub.



2. Encuentra el operador de la comunidad NetApp Trident.



No selecciones **Eliminar todas las instancias de operandos de este operador**.

3. Haz clic en **Desinstalar**.
4. Después de que termine la desinstalación, sigue con [Instala el operador certificado OpenShift](#).

Cambia desde una instalación de operador basada en Helm

Pasos

1. Muestra la versión de Helm para tu instalación de Trident:

```
helm ls -n trident
```

2. Desinstala la versión de Helm:

```
helm uninstall <release-name> -n trident
```

3. Después de que termine la desinstalación, sigue con [Instala el operador certificado OpenShift](#).

Cambiar desde un operador desplegado manualmente

Si instalaste Trident desplegando manualmente el operador usando un `bundle.yaml` del paquete de instalación, elimínalo borrando el mismo manifiesto.

Pasos

1. Elimina el despliegue del operador usando el manifiesto del bundle:

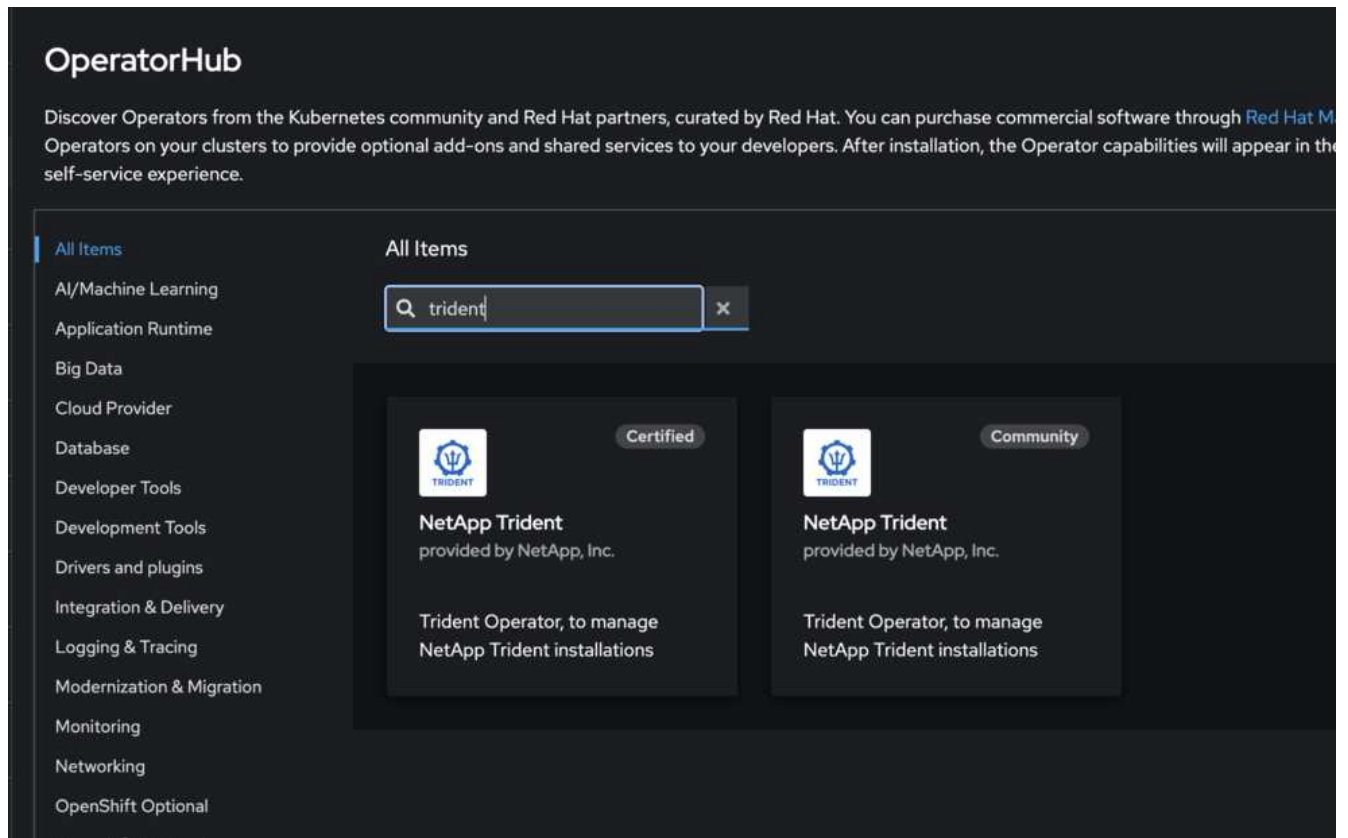
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Después de que termine la desinstalación, sigue con [Instala el operador certificado OpenShift](#).

Instala el operador certificado OpenShift

Pasos

1. Ve a Red Hat OperatorHub.
2. Busca y selecciona el operador NetApp Trident.



3. Sigue las instrucciones en pantalla para instalar el operador.

Verificación

- Revisa el OperatorHub en la consola para asegurarte de que el nuevo operador certificado se haya instalado correctamente.

Información de copyright

Copyright © 2026 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en <http://www.netapp.com/TM> son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.