Manos a la obra

Astra Trident

NetApp
April 16, 2024

Tabla de contenidos

Manos a la obra

Pruébelo

NetApp proporciona una imagen de laboratorio lista para usar a la que puede solicitar "Versión de prueba de NetApp". La unidad de prueba ofrece un entorno de filtrado que incluye un clúster de Kubernetes de tres nodos y una configuración de Astra Trident instalada y configurada. Es una excelente forma de familiarizarse con Astra Trident y explorar sus funciones.

Otra opción es ver la "Guía de instalación de Kubeadm" Proporcionado por Kubernetes.

> No debe usar el clúster de Kubernetes que cree con las siguientes instrucciones en producción. Use las guías de puesta en marcha de producción que ofrece su distribución para crear clústeres listos para la producción.

Si es la primera vez que utiliza Kubernetes, familiarícese con los conceptos y las herramientas "aquí".

Requisitos

Para comenzar, revise los front-ends, los back-ends y la configuración de host compatibles.

> Para obtener más información sobre los puertos que utiliza Astra Trident, consulte "aquí".

Front-ends compatibles (orquestadores)

Astra Trident admite varios orquestadores y motores de contenedor, incluidos los siguientes:

- Kubernetes 1.17 o posterior (último: 1.22)
- Mirantis Kubernetes Engine 3.4
- OpenShift 4.7, 4.8, 4.9 (último 4.9)

El operador Trident es compatible con las siguientes versiones:

- Kubernetes 1.17 o posterior (último: 1.22)
- OpenShift 4.7, 4.8, 4.9 (último 4.9)

> Los usuarios de Red Hat OpenShift Container Platform pueden observar que el archivo initiatorname.iscsi está en blanco si se utiliza una versión inferior a 4.6.8. Este es un error que ha sido identificado por RedHat para ser corregido con OpenShift 4.6.8. Vea esto "aviso de corrección de errores". NetApp le recomienda que utilice Astra Trident en OpenShift 4.7 y posteriores.

Astra Trident también funciona con una gran variedad de ofertas de Kubernetes totalmente gestionadas y gestionadas, como Google Kubernetes Engine (GKE) de Google Cloud, Elastic Kubernetes Services (EKS) de AWS, Azure Kubernetes Service (AKS) y Rancher.

Back-ends compatibles (almacenamiento)

Para utilizar Astra Trident, se necesitan uno o varios de los siguientes back-ends compatibles:

- Amazon FSX para ONTAP de NetApp
- Azure NetApp Files
- Almacén de datos Astra
- Cloud Volumes ONTAP
- Cloud Volumes Service para AWS
- Cloud Volumes Service para GCP
- FAS/AFF/Seleccione 9.3 o posterior
- Cabina All SAN de NetApp (ASA)
- Software HCI/Element de NetApp 8 o posterior

Requisitos de funciones

La siguiente tabla resume las funciones disponibles con esta versión de Astra Trident y las versiones de Kubernetes compatible.

Función	La versión de Kubernetes	¿Se requieren puertas de funciones?
CSI Trident	1.17 y posterior	No
Snapshots de volumen	1.17 y posterior	No
RVP desde snapshots de volumen	1.17 y posterior	No
Cambio de tamaño del VP de iSCSI	1.17 y posterior	No
CHAP bidireccional de ONTAP	1.17 y posterior	No
Políticas de exportación dinámicas	1.17 y posterior	No
Operador de Trident	1.17 y posterior	No
Preparación automática de nodos de trabajo (beta)	1.17 y posterior	No
Topología CSI	1.17 y posterior	No

Se probaron sistemas operativos host

De forma predeterminada, Astra Trident se ejecuta en un contenedor y, por lo tanto, se ejecutará en cualquier trabajador de Linux. Sin embargo, estos trabajadores deben poder montar los volúmenes que ofrece Astra Trident con el cliente NFS o iniciador iSCSI estándar, en función de los back-ends que utilice.

Aunque Astra Trident no «admite» oficialmente sistemas operativos específicos, se sabe que las siguientes distribuciones de Linux funcionan:

- Redhat CoreOS 4.2 y 4.3

- RHEL o CentOS 7.4 o posterior
- Ubuntu 18.04 o posterior

La `tridentctl` Utility también se ejecuta en cualquiera de estas distribuciones de Linux.

Configuración de hosts

En función de los back-end que se estén utilizando, deben instalarse utilidades NFS y/o iSCSI en todos los trabajadores del clúster. Consulte "aquí" si quiere más información.

Configuración del sistema de almacenamiento

Es posible que Astra Trident requiera algunos cambios en un sistema de almacenamiento antes de que una configuración de back-end pueda usarlos. Consulte "aquí" para obtener más detalles.

Imágenes de contenedor y las versiones de Kubernetes correspondientes

Para instalaciones con problemas de conexión aérea, la siguiente lista es una referencia de las imágenes de contenedor necesarias para instalar Astra Trident. Utilice la `tridentctl images` comando para verificar la lista de imágenes de contenedor necesarias.

La versión de Kubernetes	Imagen de contenedor
v1.17.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)
v1.18.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)

La versión de Kubernetes	Imagen de contenedor
v1.19.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v2.2 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)
v1.20.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v3.0.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)
1.21.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v3.0.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)

La versión de Kubernetes	Imagen de contenedor
v1.22.0	• netapp/trident:21.10.0 • netapp/trident-autosupport:21.10 • k8s.gcr.io/sig-storage/csi-aprovisionador:v3.0.0 • k8s.gcr.io/sig-storage/csi-attacher:v3.3.0 • k8s.gcr.io/sig-storage/csi-reszer:v1.3.0 • k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3 • k8s.gcr.io/sig-storage/csi-node-driver-registrador:v2.3.0 • netapp/operador especializado: 21.10.0 (opcional)

En la versión 1.20 de Kubernetes y versiones posteriores, utilice las validadas `k8s.gcr.io/sig-storage/csi-snapshotter:v4.x` la imagen sólo si la `v1` la versión sirve `volumesnapshots.snapshot.storage.k8s.io` CRD. Si la `v1beta1` La versión sirve al CRD con/sin el `v1` versión, utilice la validada `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` imagen.

Información general sobre la implementación

Puede poner en marcha Astra Trident con el operador de Trident o con `tridentctl`.

Elija el método de implementación

Para determinar qué método de implementación utilizar, tenga en cuenta lo siguiente:

¿Por qué debo usar el operador Trident?

La "Operador de Trident" Es una excelente forma de gestionar de forma dinámica los recursos de Astra Trident y automatizar la fase de configuración. Hay algunos requisitos previos que deben cumplirse. Consulte "los requisitos".

El operador de Trident ofrece varias ventajas como se describe a continuación.

Funcionalidad de reparación automática

Puede supervisar una instalación de Astra Trident y tomar medidas de forma activa para solucionar problemas, como, por ejemplo, cuándo se elimina la implementación o si se modifica por accidente. Cuando el operador se configura como una implementación, un `trident-operator-<generated-id>` se crea el pod. Este pod asocia un `TridentOrchestrator` CR con una instalación de Astra Trident y siempre garantiza que sólo haya una activa `TridentOrchestrator`. En otras palabras, el operador se asegura de que sólo hay una instancia de Astra Trident en el cluster y controla su configuración, asegurándose de que la instalación es idempotente. Cuando se realizan cambios en la instalación (como eliminar el despliegue o el conjunto de nodos), el operador los identifica y los corrige individualmente.

Actualizaciones sencillas en instalaciones existentes

Puede actualizar fácilmente una implementación existente con el operador. Sólo tiene que editar el

`TridentOrchestrator` CR para realizar actualizaciones de una instalación. Por ejemplo, piense en una situación en la que necesita habilitar Astra Trident para generar registros de depuración.

Para hacer esto, parche su `TridentOrchestrator` para ajustar `spec.debug` para `true`:

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

Después `TridentOrchestrator` se actualiza, el operador procesa las actualizaciones y parches de la instalación existente. Esto puede activar la creación de nuevos POD para modificar la instalación según corresponda.

Gestiona automáticamente las actualizaciones de Kubernetes

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

> Si se actualiza el clúster a una versión no compatible, el operador evita la instalación de Astra Trident. Si ya se ha instalado Astra Trident con el operador, se muestra una advertencia para indicar que Astra Trident está instalada en una versión de Kubernetes no compatible.

¿Por qué debo usar Helm?

Si dispone de otras aplicaciones que gestiona con Helm, empezando por Astra Trident 21.01, puede gestionar su puesta en marcha también mediante Helm.

Cuándo debo usar `tridenctl`?

Si tiene una puesta en marcha existente que debe actualizarse a o si está buscando personalizar altamente su puesta en marcha, deberá echar un vistazo a utilizar "tridentctl". Este es el método convencional de puesta en marcha de Astra Trident.

Consideraciones que tener en cuenta al mover entre métodos de implementación

No es difícil imaginar un escenario en el que se desee moverse entre los métodos de implementación. Debe tener en cuenta lo siguiente antes de intentar moverse desde un `tridentctl` implementación en una implementación basada en operadores, o viceversa:

- Utilice siempre el mismo método para desinstalar Astra Trident. Si ha implementado con `tridentctl`, debe utilizar la versión adecuada de `tridentctl` Binario para desinstalar Astra Trident. Del mismo modo, si está desplegando con el operador, debe editar el `TridentOrchestrator` CR y SET `spec.uninstall=true` Para desinstalar Astra Trident.
- Si tiene una implementación basada en el operador que desea quitar y utilizar `tridentctl` Para poner en marcha Astra Trident, primero debe editar `TridentOrchestrator` y ajustar `spec.uninstall=true` Para desinstalar Astra Trident. A continuación, elimínelo `TridentOrchestrator` y la puesta en marcha del operador. A continuación, puede realizar la instalación mediante `tridentctl`.
- Si tiene una puesta en marcha manual basada en el operador y desea utilizar la puesta en marcha del operador de Trident basado en Helm, primero debe desinstalar manualmente al operador y, a continuación, realizar la instalación de Helm. De este modo, Helm puede poner en marcha el operador

Trident con las etiquetas y anotaciones necesarias. Si no lo hace, la puesta en marcha del operador de Trident basado en Helm generará un error de validación de la etiqueta y un error de validación de la anotación. Si usted tiene un `tridentctl`La implementación basada en , puede utilizar la puesta en marcha basada en Helm sin que se produzcan problemas.

Entender los modos de implementación

Existen tres formas de poner en marcha Astra Trident.

Implementación estándar

La puesta en marcha de Trident en un clúster de Kubernetes se traduce en que el instalador de Astra Trident realiza dos acciones:

- Obteniendo las imágenes del contenedor a través de Internet
- Crear una implementación y/o un conjunto de nodos demonset, que hace girar Astra Trident pods en todos los nodos elegibles del clúster de Kubernetes.

Una puesta en marcha estándar como esta se puede realizar de dos formas distintas:

- Uso `tridentctl install`
- Utilice el operador Trident. Es posible poner en marcha el operador de Trident de forma manual o mediante Helm.

Este modo de instalación es la forma más sencilla de instalar Astra Trident y funciona para la mayoría de los entornos que no imponen restricciones de red.

Puesta en marcha sin conexión

Para realizar una implementación con conexión inalámbrica, puede utilizar `--image-registry` marque al invocar `tridentctl install` para apuntar a un registro de imágenes privado. Si realiza la implementación con el operador de Trident, también es posible especificar otra opción `spec.imageRegistry` en la `TridentOrchestrator`. Este registro debe contener la "Imagen de Trident", la "Imagen de Trident AutoSupport", Y las imágenes del sidecar CSI según lo requiera su versión Kubernetes.

Para personalizar su puesta en marcha, puede usar `tridentctl` Para generar los manifiestos para los recursos de Trident. Esto incluye la implementación, el conjunto demoníaco, la cuenta de servicio y el rol de clúster que crea Astra Trident como parte de su instalación.

Consulte estos enlaces para obtener más información sobre cómo personalizar la implementación:

- "Personalice la implementación basada en operador"
  *

> Si está utilizando un repositorio de imágenes privado, debe agregar `/k8scsi` Para las versiones de Kubernetes anteriores a la 1.17 o. `/sig-storage` Para las versiones de Kubernetes posteriores a la 1.17 al final de la URL del registro privado. Cuando se utiliza un registro privado para `tridentctl` implementación, debe usar `--trident-image` y.. `--autosupport-image` en conjunto con `--image-registry`. Si va a poner en marcha Astra Trident con el operador Trident, asegúrese de que orchestrator CR incluya `tridentImage` y.. `autosupportImage` en los parámetros de instalación.

Puesta en marcha remota

Aquí encontrará una descripción de alto nivel del proceso de implementación remota:

- Despliegue la versión adecuada de `kubectl` En la máquina remota desde la que desea poner en marcha Astra Trident.
- Copie los archivos de configuración del clúster de Kubernetes y establezca el `KUBECONFIG` variable de entorno en el equipo remoto.
- Inicie un `kubectl get nodes` Comando para verificar que puede conectarse al clúster de Kubernetes necesario.
- Complete la implementación desde la máquina remota mediante los pasos de instalación estándar.

Realice puestas en marcha con el operador de Trident

Puede poner en marcha Astra Trident con el operador de Trident. Es posible poner en marcha el operador de Trident de forma manual o mediante Helm.

> Si usted no se ha familiarizado ya con el "conceptos básicos", ahora es un gran momento para hacerlo.

Lo que necesitará

Para poner en marcha Astra Trident, se deben cumplir los siguientes requisitos previos:

- Tiene privilegios completos para un clúster de Kubernetes compatible que ejecute Kubernetes 1.17 y versiones posteriores.
- Tiene acceso a un sistema de almacenamiento de NetApp compatible.
- Puede montar volúmenes de todos los nodos de trabajo de Kubernetes.
- Tiene un host Linux con `kubectl` (o. `oc`, Si está utilizando OpenShift) instalado y configurado para administrar el clúster de Kubernetes que desea utilizar.
- Ha configurado el `KUBECONFIG` Variable de entorno para señalar la configuración del clúster de Kubernetes.
- Habilitó el "Puertas de funciones requeridas por Astra Trident".
- Si utiliza Kubernetes con Docker Enterprise, "Siga sus pasos para habilitar el acceso a la CLI".

¿Tiene todo eso? Estupendo. Empecemos:

Ponga en marcha el operador de Trident mediante Helm

Realice los pasos que se enumeran para implementar el operador de Trident mediante Helm.

Lo que necesitará

Además de los requisitos previos mencionados anteriormente, para poner en marcha el operador de Trident con Helm, es necesario lo siguiente:

- Kubernetes 1.17 y posteriores
- Versión timón 3

Pasos

1. Descargue el paquete del instalador de desde la "Trident GitHub" página. El paquete de instalación incluye el gráfico Helm en la `/helm` directorio.
2. Utilice la `helm install` y especifique un nombre para la implementación. Consulte el siguiente ejemplo:

   ```
   helm install <name> trident-operator-21.07.1.tgz --namespace <namespace
   you want to use for Trident>
   ```

   Si todavía no creó un espacio de nombres para Trident, puede añadir el `--create-namespace` parámetro de la `helm install` comando. Helm creará automáticamente el espacio de nombres para usted.

   Existen dos formas de pasar los datos de configuración durante la instalación:

   - `--values` (o. `-f`): Especifique un archivo YAML con anulaciones. Esto se puede especificar varias veces y el archivo de la derecha tendrá prioridad.
   - `--set`: Especifique anulaciones en la línea de comandos.

Por ejemplo, para cambiar el valor predeterminado de `debug`, ejecute lo siguiente `--set` comando:

```
$ helm install <name> trident-operator-21.07.1.tgz --set tridentDebug=true
```

La `values.yaml` Archivo, que forma parte del gráfico Helm, proporciona la lista de claves y sus valores predeterminados.

`helm list` muestra detalles sobre la instalación, como nombre, espacio de nombres, gráfico, estado, versión de la aplicación, número de revisión, etc.

Ponga en marcha manualmente el operador de Trident

Realice los pasos que se enumeran para implementar manualmente el operador de Trident.

Paso 1: Califique su clúster de Kubernetes

Lo primero que debe hacer es iniciar sesión en el host Linux y comprobar que está gestionando un *working*, "Clúster de Kubernetes compatible" que tenga los privilegios necesarios para.

> Con OpenShift, utilícelo `oc` en lugar de `kubectl` en todos los ejemplos que siguen, e inicie sesión como **system:admin** primero ejecutando `oc login -u system:admin` o. `oc login -u kube-admin`.

Para ver si la versión de Kubernetes es posterior a la 1.17, ejecute el siguiente comando:

```
kubectl version
```

Para ver si tiene privilegios de administrador de clúster Kubernetes, ejecute el siguiente comando:

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

Para verificar si puede iniciar un pod que utiliza una imagen desde Docker Hub y llegar al sistema de almacenamiento a través de la red de pod, ejecute el siguiente comando:

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

Paso 2: Descargue y configure el operador

> A partir de 21.01, el operador de Trident se limita al clúster. Para usar el operador de Trident para la instalación de Trident, se debe crear el `TridentOrchestrator` Definición de recursos personalizados (CRD) y definición de otros recursos. Debe realizar estos pasos para configurar el operador antes de poder instalar Astra Trident.

1. Descargue la última versión de "Paquete de instalación de Trident" De la sección *Downloads* y extráigalo.

   ```
   wget https://github.com/NetApp/trident/releases/download/v21.04/trident-
   installer-21.04.tar.gz
   tar -xf trident-installer-21.04.tar.gz
   cd trident-installer
   ```

2. Utilice el manifiesto CRD adecuado para crear `TridentOrchestrator` CRD. A continuación, cree un `TridentOrchestrator` Recursos personalizados más adelante para crear una instancia de la instalación por parte del operador.

   Ejecute el siguiente comando:

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

3. Después del `TridentOrchestrator` Cree CRD, cree los siguientes recursos necesarios para la implementación del operador:

   - Una cuenta de servicio para el operador
   - Una función de clúster y ClusterRoleBinding a la cuenta de servicio
   - Una política de seguridad dedicada
   - El propio operador

     El instalador de Trident contiene manifiestos para definir estos recursos. De forma predeterminada, el operador se implementa en la `trident` espacio de nombres. Si la `trident` el espacio de nombres no existe; utilice el manifiesto siguiente para crear uno.

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

4. Para desplegar el operador en un espacio de nombres distinto del predeterminado `trident` namespace, debe actualizar el `serviceaccount.yaml`, `clusterrolebinding.yaml` y.. `operator.yaml` manifiesta y genera tu `bundle.yaml`.

   Ejecute el siguiente comando para actualizar los manifiestos de YAML y generar el `bundle.yaml` con el `kustomization.yaml`:

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

   Ejecute el comando siguiente para crear los recursos e implementar el operador:

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

5. Para verificar el estado del operador después de la implementación, haga lo siguiente:

   ```
   $ kubectl get deployment -n <operator-namespace>
   NAME                READY   UP-TO-DATE   AVAILABLE   AGE
   trident-operator    1/1     1            1           3m

   $ kubectl get pods -n <operator-namespace>
   NAME                                READY   STATUS             RESTARTS
   AGE
   trident-operator-54cb664d-lnjxh   1/1     Running            0
   3m
   ```

La implementación del operador crea correctamente un pod que se ejecuta en uno de los nodos de trabajo del clúster.

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

Paso 3: Crear `TridentOrchestrator` E instale Trident

Ahora está listo para instalar Astra Trident con el operador. Esto requerirá crear `TridentOrchestrator`. El instalador de Trident incluye definiciones de ejemplo para su creación `TridentOrchestrator`. Esto inicia una instalación en `trident` espacio de nombres.

```
$ kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

$ kubectl describe torc trident
Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

El operador Trident le permite personalizar la manera en que se instala Astra Trident mediante los atributos del `TridentOrchestrator` espec. Consulte "Personalice su implementación de Trident".

El estado de `TridentOrchestrator` Indica si la instalación se realizó correctamente y muestra la versión de Trident instalada.

Estado	Descripción
Instalación	El operador está instalando Astra Trident con este método `TridentOrchestrator` CR.
Instalado	Astra Trident se ha instalado correctamente.
Desinstalando	El operador está desinstalando Astra Trident, porque `spec.uninstall=true`.
Desinstalado	Astra Trident se desinstala.
Error	El operador no pudo instalar, aplicar parches, actualizar o desinstalar Astra Trident; el operador intentará recuperarse automáticamente de este estado. Si este estado continúa, necesitará solucionar problemas.
Actualizando	El operador está actualizando una instalación existente.
Error	La `TridentOrchestrator` no se utiliza. Otro ya existe.

Durante la instalación, el estado de `TridentOrchestrator` cambios de `Installing` para `Installed`. Si observa la `Failed` y el operador no puede recuperar por sí solo, debe comprobar los registros del operador. Consulte "resolución de problemas" sección.

Puede confirmar si la instalación de Astra Trident se ha completado examinando los pods que se han creado:

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0           1m
trident-csi-mr6zc                   2/2     Running   0           1m
trident-csi-xrp7w                   2/2     Running   0           1m
trident-csi-zh2jt                   2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0           3m
```

También puede utilizar `tridentctl` Para comprobar la versión de Astra Trident instalada.

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Ahora puede Adelante y crear un back-end. Consulte "tareas posteriores a la implementación".

Para obtener información sobre la solución de problemas durante la implementación, consulte "resolución de problemas" sección.

Personalice la implementación del operador de Trident

El operador Trident le permite personalizar la manera en que se instala Astra Trident mediante los atributos del `TridentOrchestrator` espec.

Consulte la tabla siguiente para ver la lista de atributos:

Parámetro	Descripción	Predeterminado
`namespace`	Espacio de nombres para instalar Astra Trident en	"predeterminado"
`debug`	Habilite la depuración para Astra Trident	falso
`IPv6`	Instale Astra Trident sobre IPv6	falso
`k8sTimeout`	Tiempo de espera para las operaciones de Kubernetes	30 seg
`silenceAutosupport`	No envíe paquetes AutoSupport a NetApp automáticamente	falso
`enableNodePrep`	Administrar automáticamente las dependencias del nodo de trabajo (**BETA**)	falso
`autosupportImage`	La imagen contenedora para telemetría AutoSupport	"netapp/trident-autosupport:21.04.0"
`autosupportProxy`	La dirección/puerto de un proxy para enviar telemetría AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Una Marca utilizada para desinstalar Astra Trident	falso
`logFormat`	Formato de registro de Astra Trident para utilizar [text,json]	"texto"
`tridentImage`	Imagen de Astra Trident para instalar	"netapp/trident:21.04"
`imageRegistry`	Ruta de acceso al registro interno, del formato `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.17+) o quay.io/k8scsi"
`kubeletDir`	Ruta al directorio kubelet del host	“/var/lib/kubelet”
`wipeout`	Una lista de recursos para eliminar y realizar una eliminación completa de Astra Trident	

Parámetro	Descripción	Predeterminado
`imagePullSecrets`	Secretos para extraer imágenes de un registro interno	

`spec.namespace` se especifica en `TridentOrchestrator` Para indicar en qué espacio de nombres está instalado Astra Trident. Este parámetro **no se puede actualizar después de instalar Astra Trident**. Intentar hacerlo provoca el estado de `TridentOrchestrator` para cambiar a. `Failed`. Astra Trident no pretende migrar entre espacios de nombres.

La preparación automática del nodo de trabajo es una **función beta** que se utiliza únicamente en entornos no productivos.

Puede utilizar los atributos mencionados anteriormente al definir `TridentOrchestrator` para personalizar la instalación. Veamos un ejemplo:

```
$ cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: netapp/trident:21.04.0
  imagePullSecrets:
  - thisisasecret
```

Si desea personalizar la instalación más allá de lo que desee `TridentOrchestrator` los argumentos permiten, debe considerar utilizar `tridentctl` Para generar manifiestos YAML personalizados que puede modificar según sea necesario.

Despliegue con trimentctl

Puede poner en marcha Astra Trident con `tridentctl`.

Si usted no se ha familiarizado ya con el "conceptos básicos", ahora es un gran momento para hacerlo.

Para personalizar la puesta en marcha, consulte "aquí".

Lo que necesitará

Para poner en marcha Astra Trident, se deben cumplir los siguientes requisitos previos:

- Tiene privilegios completos en un clúster de Kubernetes compatible.
- Tiene acceso a un sistema de almacenamiento de NetApp compatible.

- Puede montar volúmenes de todos los nodos de trabajo de Kubernetes.
- Tiene un host Linux con `kubectl` (o. `oc`, Si está utilizando OpenShift) instalado y configurado para administrar el clúster de Kubernetes que desea utilizar.
- Ha configurado el `KUBECONFIG` Variable de entorno para señalar la configuración del clúster de Kubernetes.
- Habilitó el "Puertas de funciones requeridas por Astra Trident".
- Si utiliza Kubernetes con Docker Enterprise, "Siga sus pasos para habilitar el acceso a la CLI".

¿Tiene todo eso? Estupendo. Empecemos:

> Para obtener información acerca de cómo personalizar la implementación, consulte "aquí".

Paso 1: Califique su clúster de Kubernetes

Lo primero que debe hacer es iniciar sesión en el host Linux y comprobar que está gestionando un *working*, "Clúster de Kubernetes compatible" que tenga los privilegios necesarios para.

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

Para comprobar la versión de Kubernetes, ejecute el siguiente comando:

```
kubectl version
```

Para ver si tiene privilegios de administrador de clúster Kubernetes, ejecute el siguiente comando:

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

Para verificar si puede iniciar un pod que utiliza una imagen desde Docker Hub y llegar al sistema de almacenamiento a través de la red de pod, ejecute el siguiente comando:

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

Identifique la versión del servidor Kubernetes. Lo utilizará cuando instale Astra Trident.

Paso 2: Descargue y extraiga el instalador

> El instalador de Trident crea un pod Trident, configura los objetos CRD que se utilizan para mantener su estado e inicializa los sidecs CSI que realizan acciones, como el aprovisionamiento y la asociación de volúmenes a los hosts del clúster.

Puede descargar la última versión de "Paquete de instalación de Trident" En la sección *Downloads* y extráigalo.

Por ejemplo, si la última versión es 21.07.1:

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

Paso 3: Instalar Astra Trident

Instale Astra Trident en el espacio de nombres deseado ejecutando `tridentctl install` comando.

```
$ ./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

Se verá así cuando el instalador haya finalizado. Según el número de nodos del clúster de Kubernetes, puede observar más pods:

```
$ kubectl get pod -n trident
NAME                              READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx      4/4     Running   0          5m29s
trident-csi-vgc8n                 2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

Si ve un resultado similar al ejemplo anterior, ha completado este paso, pero Astra Trident aún no está completamente configurado. Adelante y continúe con el paso siguiente. Consulte "tareas posteriores a la implementación".

Sin embargo, si el instalador no se completa correctamente o no ve una **en ejecución** `trident-csi-<generated id>`, la plataforma no estaba instalada.

> Para obtener información sobre la solución de problemas durante la implementación, consulte "resolución de problemas" sección.

Personalice la implementación trimentctl

El instalador de Trident le permite personalizar atributos. Por ejemplo, si ha copiado la imagen de Trident en un repositorio privado, puede especificar el nombre de la imagen mediante `--trident-image`. Si ha copiado la imagen Trident así como las imágenes sidecar CSI necesarias en un repositorio privado, puede que sea preferible especificar la ubicación de ese repositorio mediante el `--image-registry` switch, que toma la forma `<registry FQDN>[:port]`.

Para que Astra Trident configure automáticamente los nodos de trabajo, utilice `--enable-node-prep`. Para obtener más información sobre cómo funciona, consulte "aquí".

> La preparación automática de nodos de trabajo es una **función beta** que se utiliza únicamente en entornos que no son de producción.

Si utiliza una distribución de Kubernetes, donde `kubelet` mantiene los datos en una ruta distinta de la habitual `/var/lib/kubelet`, puede especificar la ruta alternativa mediante `--kubelet-dir`.

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

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`

- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`

Después de haber generado estos archivos, puede modificarlos según sus necesidades y luego usarlos `--use-custom-yaml` para instalar su implementación personalizada.

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

¿Cuál es el siguiente?

Después de implementar Astra Trident, puede continuar con la creación de un entorno de administración, la creación de una clase de almacenamiento, el aprovisionamiento de un volumen y el montaje del volumen en un pod.

Paso 1: Crear un back-end

Ahora puede Adelante y crear un back-end que utilizará Astra Trident para aprovisionar volúmenes. Para ello, cree un `backend.json` archivo que contiene los parámetros necesarios. Se pueden encontrar archivos de configuración de ejemplo para diferentes tipos de backend en la `sample-input` directorio.

Consulte "aquí" para obtener más información acerca de cómo configurar el archivo para el tipo de backend.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                  UUID                |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

Si la creación falla, algo estaba mal en la configuración del back-end. Puede ver los registros para determinar la causa ejecutando el siguiente comando:

```
./tridentctl -n trident logs
```

Después de solucionar el problema, simplemente vuelva al principio de este paso e inténtelo de nuevo. Para obtener más consejos sobre la solución de problemas, consulte "la solución de problemas" sección.

Paso 2: Crear una clase de almacenamiento

Los usuarios de Kubernetes aprovisionan volúmenes mediante reclamaciones de volumen persistente (RVP) que especifican un "clase de almacenamiento" por nombre. Los detalles están ocultos de los usuarios, pero una clase de almacenamiento identifica el aprovisionador que se utiliza para esa clase (en este caso, Trident) y lo que significa esa clase para el aprovisionador.

Cree una clase de almacenamiento que los usuarios de Kubernetes especifiquen cuando quieran un volumen. La configuración de la clase debe modelar el back-end que ha creado en el paso anterior, de modo que Astra Trident lo utilice para aprovisionar nuevos volúmenes.

La clase de almacenamiento más sencilla que se debe empezar por está basada en la `sample-input/storage-class-csi.yaml.templ` archivo que viene con el instalador, reemplazar *`BACKEND_TYPE`* con el nombre del controlador de almacenamiento.

```
./tridentctl -n trident get backend
+-------------+----------------+-------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                |
STATE  | VOLUMES |
+-------------+----------------+-------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+-------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Este es un objeto de Kubernetes, por lo que se usa `kubectl` Para crear en Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Ahora debería ver una clase de almacenamiento * Basic-csi* tanto en Kubernetes como en Astra Trident, y Astra Trident debería haber descubierto las piscinas en el back-end.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Paso 3: Aprovisionar el primer volumen

Ahora está listo para aprovisionar de forma dinámica el primer volumen. Esto se realiza mediante la creación de un Kubernetes "reclamación de volumen persistente" Objeto (PVC).

Cree una RVP para un volumen que utiliza la clase de almacenamiento que acaba de crear.

Consulte `sample-input/pvc-basic-csi.yaml` por ejemplo. Asegúrese de que el nombre de la clase de almacenamiento coincida con el que ha creado.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Paso 4: Monte los volúmenes en un pod

Ahora vamos a montar el volumen. Lanzaremos una vaina nginx que monta el PV debajo `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

En este momento, el pod (la aplicación) ya no existe pero el volumen sigue ahí. Puede utilizarlo desde otro pod si lo desea.

Para eliminar el volumen, elimine la reclamación:

```
kubectl delete pvc basic
```

Ahora puede realizar tareas adicionales, como las siguientes:

- "Configurar back-ends adicionales."
- "Cree clases de almacenamiento adicionales."