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".

Obtenga más información sobre la versión de prueba

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

Antes de instalar Astra Trident, debería revisar estos requisitos generales del sistema. Es posible que los back-ends específicos tengan requisitos adicionales.

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

** información bï¿½tico sobre Astra Tridbï¿½tico **

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Front-ends compatibles (orquestadores)

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

- Anthos en las instalaciones (VMware) y Anthos en bare metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- Mirantis Kubernetes Engine 3.5

- OpenShift 4.9 - 4.12

El operador Trident es compatible con las siguientes versiones:

- Anthos en las instalaciones (VMware) y Anthos en bare metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- OpenShift 4.9 - 4.12

Astra Trident también funciona con una gran cantidad de otras ofertas de Kubernetes completamente gestionadas y gestionadas, como Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher y VMware Tanzu Portfolio.

Antes de actualizar un clúster de Kubernetes de 1.24 a 1.25 o posterior que tenga instalado Astra Trident, consulte "Actualice la instalación de un operador basado en Helm".

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
- Cloud Volumes ONTAP
- Cloud Volumes Service para GCP
- FAS/AFF/Seleccione 9.5 o posterior
- Cabina All SAN de NetApp (ASA)
- Software HCI/Element de NetApp 11 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	1.21 - 1.26	No
Snapshots de volumen	1.21 - 1.26	No
RVP desde snapshots de volumen	1.21 - 1.26	No
Cambio de tamaño del VP de iSCSI	1.21 - 1.26	No
CHAP bidireccional de ONTAP	1.21 - 1.26	No
Políticas de exportación dinámicas	1.21 - 1.26	No

Función	La versión de Kubernetes	¿Se requieren puertas de funciones?
Operador de Trident	1.21 - 1.26	No
Topología CSI	1.21 - 1.26	No

Se probaron sistemas operativos host

Aunque Astra Trident no admite oficialmente sistemas operativos específicos, se sabe que lo siguiente es lo siguiente:

- Versiones de RedHat CoreOS (RHCOS) compatibles con OpenShift Container Platform
- RHEL 8 O POSTERIOR
- Ubuntu 22.04 o posterior
- Windows Server 2019

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.

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

Configuración de hosts

Todos los nodos de trabajadores del clúster de Kubernetes deben poder montar los volúmenes que haya aprovisionado para los pods. Para preparar los nodos de trabajo, debe instalar las herramientas NFS o iSCSI según su selección de controlador.

"Prepare el nodo de trabajo"

Configuración del sistema de almacenamiento

Es posible que Astra Trident requiera cambios en un sistema de almacenamiento antes de que una configuración de back-end pueda usarla.

"Configurar los back-ends"

Puertos Astra Trident

Astra Trident requiere acceso a puertos específicos para la comunicación.

"Puertos Astra Trident"

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
1.21.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)
v1.22.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)
v1.23.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)

La versión de Kubernetes	Imagen de contenedor
v1.24.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)
v1.25.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)
v1.26.0	• docker.io/netapp/trident:23.01.1 • docker.io/netapp/trident-autosupport:23,01 • registry.k8s.io/sig-storage/csi-provisioner:v3,4.0 • registry.k8s.io/sig-storage/csi-attacher:v4,1.0 • registry.k8s.io/sig-storage/csi-resizer:v1,7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6,2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrador:v2,7.0 • docker.io/netapp/trident-operator:23.01.1 (opcional)

En la versión 1.21 de Kubernetes y versiones posteriores, utilice la validada `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` la imagen sólo si la `v1` la versión sirve `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD. Si la `v1beta1` La versión sirve al CRD con/sin el `v1` versión, utilice la validada `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` imagen.

Instale Astra Trident

Obtenga más información sobre la instalación de Astra Trident

Para garantizar que Astra Trident se puede instalar en una amplia variedad de entornos y organizaciones, NetApp ofrece múltiples opciones de instalación. Puede instalar Astra Trident con el operador Trident (manualmente o mediante Helm) o con `tridentctl`. En este tema se proporciona información importante para seleccionar el proceso de instalación adecuado.

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

información bï¿½tico sobre Astra Tridbï¿½tico

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

  Uso de la configuración sin multivía o el uso de `find_multipaths: yes` o. `find_multipaths: smart` el valor del 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 la ruta de instalación, debe tener:

- Privilegios completos en un clúster de Kubernetes compatible que ejecuta una versión compatible de Kubernetes y requisitos de funciones habilitados. Revise la "requisitos" para obtener más detalles.
- Acceso a un sistema de almacenamiento de NetApp compatible.
- Capacidad para montar volúmenes de todos los nodos de trabajo de Kubernetes.
- Un host Linux con `kubectl` (o. `oc`, Si está utilizando OpenShift) instalado y configurado para administrar el clúster de Kubernetes que desea utilizar.
- La `KUBECONFIG` Variable de entorno establecida en el clúster de Kubernetes.
- Si utiliza Kubernetes con Docker Enterprise, "Siga sus pasos para habilitar el acceso a la CLI".

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

Elija el método de instalación

Seleccione el método de instalación adecuado. También debe revisar las consideraciones de "moverse entre métodos" antes de tomar su decisión.

Utilice el operador Trident

Tanto si se pone en marcha manualmente como si se utiliza Helm, el operador Trident es una forma excelente de simplificar la instalación y gestionar de forma dinámica los recursos de Astra Trident. Incluso puede "Personalice la implementación del operador de Trident" uso de los atributos de la `TridentOrchestrator` Recurso personalizado (CR).

Algunas de las ventajas de usar el operador Trident son:

** Astra Trident de objetos cremunidad **

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

- ServiceAccount para el operador
- ClusterRole y ClusterRoleBinding a la cuenta de servicio
- Dedicated PodSecurityPolicy (para Kubernetes 1.25 y versiones anteriores)
- El propio operador

** de capeel de curación de las Ouna**

El operador supervisa la instalación de Astra Trident y toma activamente medidas para resolver problemas, como cuándo se elimina la implementación o si se modifica accidentalmente. A. `trident-operator-<generated-id>` se crea un pod que asocia un `TridentOrchestrator` CR con una instalación de Astra Trident. Esto garantiza que solo haya una instancia de Astra Trident en el clúster y controle 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 en la existente

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.

**Mejora a de Kubernetes a mano **

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.

** gestiona clústeres a través de BlueXP (anteriormente Cloud Manager)**

Con "Astra Trident con BlueXP", Puede realizar una actualización a la versión más reciente de Astra Trident, agregar y gestionar clases de almacenamiento, conectarlos a entornos de trabajo y realizar backups de volúmenes persistentes mediante Cloud Backup Service. BlueXP admite la puesta en marcha de Astra Trident con el operador de Trident, ya sea manualmente o mediante Helm.

Uso `tridentctl`

Si tiene una puesta en marcha existente que debe actualizarse o si desea personalizar altamente su puesta en marcha, debe tener en cuenta . Este es el método convencional de puesta en marcha de Astra Trident.

Puede hacerlo Para generar los manifiestos de 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.

> A partir de la versión 22.04, las claves AES ya no se regenerarán cada vez que se instale Astra Trident. Con este lanzamiento, Astra Trident instalará un nuevo objeto secreto que persiste en todas las instalaciones. Esto significa que, `tridentctl` En la versión 22.04 puede desinstalar versiones anteriores de Trident, pero las versiones anteriores no pueden desinstalar instalaciones de 22.04. Seleccione la instalación *method* adecuada.

Elija el modo de instalación

Determine el proceso de implementación según el *installation mode* (Standard, Offline o Remote) requerido por su organización.

Instalación estándar

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

Cuando se utiliza el modo estándar, el instalador de Astra Trident:

- Obtiene las imágenes del contenedor por Internet
- Crea una implementación o un conjunto de nodos demonset, que hace girar las pods de Astra Trident en todos los nodos elegibles del clúster de Kubernetes

Instalación sin conexión

Es posible que se requiera el modo de instalación sin conexión en una ubicación segura o con un sistema de activación por aire. En este escenario, puede crear un único registro privado duplicado o dos registros reflejados para almacenar las imágenes Trident y CSI necesarias.

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

Instalación remota

A continuación se ofrece una descripción general de alto nivel del proceso de instalación 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.

Seleccione el proceso según el método y el modo

Después de tomar sus decisiones, seleccione el proceso apropiado.

Método	Modo de instalación
Operador de Trident (manualmente)	"Instalación estándar" "Instalación sin conexión"
Operador Trident (Helm)	"Instalación estándar" "Instalación sin conexión"
`tridentctl`	"Instalación estándar o sin conexión"

Moverse entre los métodos de instalación

Puede decidir cambiar el método de instalación. Antes de hacerlo, tenga en cuenta lo siguiente:

- Utilice siempre el mismo método para instalar y 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 operador que desea quitar y utilizar en su lugar `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, llevar a cabo 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.

Otras opciones de configuración conocidas

Al instalar Astra Trident en productos de la cartera tanzu de VMware:

- El clúster debe admitir cargas de trabajo con privilegios.
- La `--kubelet-dir` el indicador se debe establecer en la ubicación del directorio kubelet. De forma predeterminada, esta es `/var/vcap/data/kubelet`.

  Especificación de la ubicación del kubelet mediante `--kubelet-dir` Sabe que funciona para el operador, Helm y. `tridentctl` implementaciones.

Realice la instalación mediante el operador Trident

Implemente manualmente el operador de Trident (modo estándar)

Es posible poner en marcha manualmente el operador de Trident para instalar Astra Trident. Este proceso se aplica a instalaciones en las que las imágenes de contenedor requeridas por Astra Trident no se almacenan en un registro privado. Si dispone de un registro de imágenes privado, utilice "proceso de puesta en marcha sin conexión".

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

** información bï¿½tico sobre Astra Tridbï¿½tico **

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Implemente manualmente el operador de Trident e instale Trident

Revisar "descripción general de la instalación" para asegurarse de cumplir con los requisitos previos de instalación y seleccionar la opción de instalación correcta para el entorno.

Antes de empezar

Antes de iniciar la instalación, inicie sesión en el host Linux y compruebe que esté gestionando un trabajo, "Clúster de Kubernetes compatible" y que tenga los privilegios necesarios.

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`.

1. Compruebe su versión de Kubernetes:

```
kubectl version
```

2. Comprobar los privilegios de administrador de clúster:

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

3. Compruebe que puede iniciar un pod que utilice una imagen de Docker Hub para llegar al sistema de almacenamiento a través de la red de pod:

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

Paso 1: Descargue el paquete de instalación de Trident

El paquete de instalación de Astra Trident incluye todo lo necesario para poner en marcha al operador de Trident e instalar Astra Trident. Descargue y extraiga la versión más reciente del instalador de Trident "La sección *Assets* de GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Paso 2: Cree la `TridentOrchestrator` CRD

Cree el `TridentOrchestrator` Definición de recurso personalizado (CRD). Creará una `TridentOrchestrator` Recursos personalizados más adelante. Use la versión adecuada de CRD YAML en `deploy/crds` para crear la `TridentOrchestrator` CRD.

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

Paso 3: Ponga en marcha el operador de Trident

El instalador de Astra Trident proporciona un archivo de paquete que se puede utilizar para instalar el operador y crear objetos asociados. El archivo de paquete es una forma sencilla de poner en marcha el operador e instalar Astra Trident con una configuración predeterminada.

- Para los clústeres que ejecutan Kubernetes 1.24 o inferior, utilice `bundle_pre_1_25.yaml`.
- Utilice el para clústeres que ejecuten Kubernetes 1.25 o superior `bundle_post_1_25.yaml`.

El instalador de Trident pone en marcha el operador en la `trident` espacio de nombres. Si la `trident` no existe el espacio de nombres, utilice `kubectl apply -f deploy/namespace.yaml` para crearlo.

Pasos

1. Crear los recursos e implementar el operador:

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

   > Para implementar el operador en un espacio de nombres distinto del `trident` espacio de nombres, actualización `serviceaccount.yaml`, `clusterrolebinding.yaml` y.. `operator.yaml` y genere el archivo del paquete con el `kustomization.yaml`:
   >
   > ```
   > kubectl kustomize deploy/ > deploy/<bundle>.yaml
   > ```

2. Compruebe que el operador se ha desplegado.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

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

Paso 4: Cree el `TridentOrchestrator` E instale Trident

Ahora puede crear el `TridentOrchestrator` E instale Astra Trident. Opcionalmente, puede hacerlo "Personalice su instalación de Trident" uso de los atributos de la `TridentOrchestrator` espec.

```
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:23.01
    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:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Compruebe la instalación

Existen varias formas de verificar su instalación.

Uso `TridentOrchestrator` estado

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

Trident instalada. 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, "compruebe los registros".

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.

Uso del estado de creación de pod

Para confirmar si la instalación de Astra Trident ha finalizado, revise 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
```

Uso `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

El futuro

Ahora es posible "cree una clase de back-end y almacenamiento, aprovisione un volumen y monte el volumen en un pod".

Implemente manualmente el operador Trident (modo sin conexión).

Es posible poner en marcha manualmente el operador de Trident para instalar Astra Trident. Este proceso se aplica a instalaciones en las que las imágenes de contenedor requeridas por Astra Trident se almacenan en un registro privado. Si no dispone de un registro de imágenes privado, utilice "proceso de implementación estándar".

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

información bï¿½tico sobre Astra Tridbï¿½tico

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Implemente manualmente el operador de Trident e instale Trident

Revisar "descripción general de la instalación" para asegurarse de cumplir con los requisitos previos de instalación y seleccionar la opción de instalación correcta para el entorno.

Antes de empezar

Inicie sesión en el host Linux y compruebe que está gestionando un funcionamiento y. "Clúster de Kubernetes compatible" y que tenga los privilegios necesarios.

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`.

1. Compruebe su versión de Kubernetes:

```
kubectl version
```

2. Comprobar los privilegios de administrador de clúster:

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

3. Compruebe que puede iniciar un pod que utilice una imagen de Docker Hub para llegar al sistema de almacenamiento a través de la red de pod:

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

Paso 1: Descargue el paquete de instalación de Trident

El paquete de instalación de Astra Trident incluye todo lo necesario para poner en marcha al operador de Trident e instalar Astra Trident. Descargue y extraiga la versión más reciente del instalador de Trident "La sección *Assets* de GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Paso 2: Cree la `TridentOrchestrator` CRD

Cree el `TridentOrchestrator` Definición de recurso personalizado (CRD). Creará una `TridentOrchestrator` Recursos personalizados más adelante. Use la versión adecuada de CRD YAML en `deploy/crds` para crear la `TridentOrchestrator` CRD:

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

Paso 3: Actualice la ubicación del registro en el operador

Pulg `/deploy/operator.yaml`, actualizar `image: docker.io/netapp/trident-operator:23.01.1` para reflejar la ubicación del registro de imágenes. Su "Imágenes Trident y CSI" Se pueden ubicar en un registro o en diferentes registros, pero todas las imágenes CSI deben estar ubicadas en el mismo registro. Por ejemplo:

- `image: <your-registry>/trident-operator:23.01.1` si todas las imágenes están ubicadas en un registro.

- `image: <your-registry>/netapp/trident-operator:23.01.1` Si su imagen Trident se encuentra en un registro diferente de sus imágenes CSI.

Paso 4: Ponga en marcha el operador de Trident

El instalador de Trident pone en marcha el operador en la `trident` espacio de nombres. Si la `trident` no existe el espacio de nombres, utilice `kubectl apply -f deploy/namespace.yaml` para crearlo.

Para implementar el operador en un espacio de nombres distinto del `trident` espacio de nombres, actualización `serviceaccount.yaml`, `clusterrolebinding.yaml` y.. `operator.yaml` antes de desplegar el operador.

1. Crear los recursos e implementar el operador:

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

> El instalador de Astra Trident proporciona un archivo de paquete que se puede utilizar para instalar el operador y crear objetos asociados. El archivo de paquete es una forma sencilla de poner en marcha el operador e instalar Astra Trident con una configuración predeterminada.
>
> - Para los clústeres que ejecutan Kubernetes 1.24 o inferior, utilice `bundle_pre_1_25.yaml`.
> - Utilice el para clústeres que ejecuten Kubernetes 1.25 o superior `bundle_post_1_25.yaml`.

2. Compruebe que el operador se ha desplegado.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

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

Paso 5: Actualice la ubicación del registro de imágenes en el `TridentOrchestrator`

Su "Imágenes Trident y CSI" Se pueden ubicar en un registro o en diferentes registros, pero todas las imágenes CSI deben estar ubicadas en el mismo registro. Actualizar `deploy/crds/tridentorchestrator_cr.yaml` para agregar las especificaciones de ubicación adicionales basadas en la configuración de su registro.

Imágenes en un registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

Imágenes en diferentes registros

Debe añadir `sig-storage` para la `imageRegistry` para usar diferentes ubicaciones de registro.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

Paso 6: Cree el `TridentOrchestrator` E instale Trident

Ahora puede crear el `TridentOrchestrator` E instale Astra Trident. Si lo desea, puede ir más allá "Personalice su instalación de Trident" uso de los atributos de la `TridentOrchestrator` espec. En el siguiente ejemplo se muestra una instalación donde las imágenes Trident y CSI se encuentran en diferentes registros.

```
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:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Compruebe la instalación

Existen varias formas de verificar su instalación.

Uso `TridentOrchestrator` estado

El estado de `TridentOrchestrator` Indica si la instalación se realizó correctamente y muestra la versión de Trident instalada. 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, "compruebe los registros".

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.

Uso del estado de creación de pod

Para confirmar si la instalación de Astra Trident ha finalizado, revise 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
```

Uso `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

El futuro

Ahora es posible "cree una clase de back-end y almacenamiento, aprovisione un volumen y monte el volumen en un pod".

Puesta en marcha del operador de Trident con Helm (modo estándar)

Puede poner en marcha el operador de Trident e instalar Astra Trident con Helm. Este proceso se aplica a instalaciones en las que las imágenes de contenedor requeridas por Astra Trident no se almacenan en un registro privado. Si dispone de un registro de imágenes privado, utilice "proceso de puesta en marcha sin conexión".

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

** información bï¿½tico sobre Astra Tridbï¿½tico **

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Ponga en marcha el operador de Trident e instale Astra Trident con Helm

Usar Trident "Carta del timón" Es posible poner en marcha el operador de Trident e instalar Trident en un paso.

Revisar "descripción general de la instalación" para asegurarse de cumplir con los requisitos previos de instalación y seleccionar la opción de instalación correcta para el entorno.

Antes de empezar

Además de la "requisitos previos a la implementación" que necesita "Versión timón 3".

Pasos

1. Añada el repositorio de Astra Trident Helm:

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

2. Uso `helm install` y especifique un nombre para la implementación como en el ejemplo siguiente donde `23.01.1` Es la versión de Astra Trident que está instalando.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace <trident-namespace>
```

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

Puede utilizar `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación, y el número de revisión.

Pasar los datos de configuración durante la instalación

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

Opción	Descripción
`--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 donde `23.01.1` Es la versión de Astra Trident que está instalando:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Opciones de configuración

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

Opción	Descripción	Predeterminado
`nodeSelector`	Etiquetas de nodo para la asignación de pod	
`podAnnotations`	Anotaciones del pod	
`deploymentAnnotations`	Anotaciones de despliegue	
`tolerations`	Toleraciones para la asignación de POD	
`affinity`	Afinidad para la asignación de pod	
`tridentControllerPluginNodeSelector`	Selectores de nodos adicionales para POD. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`tridentControllerPluginTolerations`	Anula la toleración de Kubernetes en pods. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`tridentNodePluginNodeSelector`	Selectores de nodos adicionales para POD. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	

Opción	Descripción	Predeterminado
`tridentNodePluginTolerations`	Anula la toleración de Kubernetes en pods. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`imageRegistry`	Identifica el registro del `trident-operator`, `trident`, y otras imágenes. Déjelo vacío para aceptar el valor predeterminado.	""
`imagePullPolicy`	Establece la política de extracción de imágenes para el `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Establece los secretos de extracción de imágenes para el `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 configurar en debug el nivel de registro del operador Trident.	`true`
`operatorImage`	Permite la sustitución completa de la imagen durante `trident-operator`.	""
`operatorImageTag`	Permite sobrescribir la etiqueta del `trident-operator` imagen.	""
`tridentIPv6`	Permite permitir que Astra Trident funcione en clústeres de 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`
`tridentHttpRequestTimeout`	Sustituye el timeout por defecto de 90 segundos para las solicitudes HTTP, con `0s` ser una duración infinita para el timeout. No se permiten valores negativos.	`"90s"`
`tridentSilenceAutosupport`	Permite deshabilitar la generación de informes periódicos de AutoSupport de Astra Trident.	`false`

Opción	Descripción	Predeterminado
`tridentAutosupportImageTag`	Permite sobrescribir la etiqueta de la imagen del contenedor AutoSupport de Astra Trident.	`<version>`
`tridentAutosupportProxy`	Permite que el contenedor Astra Trident AutoSupport telefonee a casa a través de un proxy HTTP.	""
`tridentLogFormat`	Establece el formato de registro de Astra Trident (`text` o. `json`).	`"text"`
`tridentDisableAuditLog`	Deshabilita el registro de auditorías de Astra Trident.	`true`
`tridentLogLevel`	Permite establecer el nivel de registro de Astra Trident en: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`tridentDebug`	Permite establecer el nivel de registro de Astra Trident `debug`.	`false`
`tridentLogWorkflows`	Permite habilitar flujos de trabajo específicos de Astra Trident para el registro de seguimiento o la supresión de registros.	""
`tridentLogLayers`	Permite habilitar capas específicas de Astra Trident para el registro de seguimiento o la supresión de registros.	""
`tridentImage`	Permite anular por completo la imagen de Astra Trident.	""
`tridentImageTag`	Permite sobrescribir la etiqueta de la imagen para Astra Trident.	""
`tridentProbePort`	Permite sobrescribir el puerto predeterminado utilizado para las sondas de vida/preparación de Kubernetes.	""
`windows`	Permite instalar Astra Trident en el nodo de trabajo de Windows.	`false`
`enableForceDetach`	Permite habilitar la función Forzar separación.	`false`
`excludePodSecurityPolicy`	Excluye la política de seguridad del pod del operador de la creación.	`false`

Descripción de los pods de la controladora y los pods de nodo

Astra Trident se ejecuta como un único pod de la controladora, más un pod de nodos en cada nodo de trabajo del clúster. El pod del nodo debe ejecutarse en cualquier host en el que desee montar potencialmente un volumen Astra Trident.

Kubernetes "selectores de nodos" y.. "toleraciones y tintes" se utilizan para restringir un pod para ejecutarse en un nodo concreto o preferido. Uso del "ControllerPlugin" y. `NodePlugin`, puede especificar restricciones y anulaciones.

- El complemento de la controladora se ocupa del aprovisionamiento y la gestión de volúmenes, como snapshots y redimensionamiento.
- El complemento de nodo se encarga de conectar el almacenamiento al nodo.

El futuro

Ahora es posible "cree una clase de back-end y almacenamiento, aprovisione un volumen y monte el volumen en un pod".

Implementar el operador de Trident con Helm (modo sin conexión)

Puede poner en marcha el operador de Trident e instalar Astra Trident con Helm. Este proceso se aplica a instalaciones en las que las imágenes de contenedor requeridas por Astra Trident se almacenan en un registro privado. Si no dispone de un registro de imágenes privado, utilice "proceso de implementación estándar".

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

** información bï¿½tico sobre Astra Tridbï¿½tico **

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Ponga en marcha el operador de Trident e instale Astra Trident con Helm

Usar Trident "Carta del timón" Es posible poner en marcha el operador de Trident e instalar Trident en un paso.

Revisar "descripción general de la instalación" para asegurarse de cumplir con los requisitos previos de instalación y seleccionar la opción de instalación correcta para el entorno.

Antes de empezar

Además de la "requisitos previos a la implementación" que necesita "Versión timón 3".

Pasos

1. Añada el repositorio de Astra Trident Helm:

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

2. Uso `helm install` y especifique un nombre para su implementación y ubicación del registro de imágenes. Su "Imágenes Trident y CSI" Se pueden ubicar en un registro o en diferentes registros, pero todas las imágenes CSI deben estar ubicadas en el mismo registro. En los ejemplos: `23.01.1` Es la versión de Astra Trident que está instalando.

Imágenes en un registro

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Imágenes en diferentes registros

Debe añadir `sig-storage` para la `imageRegistry` para usar diferentes ubicaciones de registro.

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

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

Puede utilizar `helm list` para revisar detalles de la instalación como nombre, espacio de nombres, gráfico, estado, versión de la aplicación, y el número de revisión.

Pasar los datos de configuración durante la instalación

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

Opción	Descripción
`--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 donde `23.01.1` Es la versión de Astra Trident que está instalando:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Opciones de configuración

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

Opción	Descripción	Predeterminado
`nodeSelector`	Etiquetas de nodo para la asignación de pod	
`podAnnotations`	Anotaciones del pod	
`deploymentAnnotations`	Anotaciones de despliegue	
`tolerations`	Toleraciones para la asignación de POD	
`affinity`	Afinidad para la asignación de pod	
`tridentControllerPluginNodeSelector`	Selectores de nodos adicionales para POD. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`tridentControllerPluginTolerations`	Anula la toleración de Kubernetes en pods. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`tridentNodePluginNodeSelector`	Selectores de nodos adicionales para POD. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`tridentNodePluginTolerations`	Anula la toleración de Kubernetes en pods. Consulte Descripción de los pods de la controladora y los pods de nodo para obtener más detalles.	
`imageRegistry`	Identifica el registro del `trident-operator`, `trident`, y otras imágenes. Déjelo vacío para aceptar el valor predeterminado.	""
`imagePullPolicy`	Establece la política de extracción de imágenes para el `trident-operator`.	`IfNotPresent`

Opción	Descripción	Predeterminado
`imagePullSecrets`	Establece los secretos de extracción de imágenes para el `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 configurar en debug el nivel de registro del operador Trident.	`true`
`operatorImage`	Permite la sustitución completa de la imagen durante `trident-operator`.	""
`operatorImageTag`	Permite sobrescribir la etiqueta del `trident-operator` imagen.	""
`tridentIPv6`	Permite permitir que Astra Trident funcione en clústeres de 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`
`tridentHttpRequestTimeout`	Sustituye el timeout por defecto de 90 segundos para las solicitudes HTTP, con `0s` ser una duración infinita para el timeout. No se permiten valores negativos.	`"90s"`
`tridentSilenceAutosupport`	Permite deshabilitar la generación de informes periódicos de AutoSupport de Astra Trident.	`false`
`tridentAutosupportImageTag`	Permite sobrescribir la etiqueta de la imagen del contenedor AutoSupport de Astra Trident.	`<version>`
`tridentAutosupportProxy`	Permite que el contenedor Astra Trident AutoSupport telefonee a casa a través de un proxy HTTP.	""
`tridentLogFormat`	Establece el formato de registro de Astra Trident (`text` o. `json`).	`"text"`
`tridentDisableAuditLog`	Deshabilita el registro de auditorías de Astra Trident.	`true`

Opción	Descripción	Predeterminado
`tridentLogLevel`	Permite establecer el nivel de registro de Astra Trident en: `trace`, `debug`, `info`, `warn`, `error`, o. `fatal`.	`"info"`
`tridentDebug`	Permite establecer el nivel de registro de Astra Trident `debug`.	`false`
`tridentLogWorkflows`	Permite habilitar flujos de trabajo específicos de Astra Trident para el registro de seguimiento o la supresión de registros.	""
`tridentLogLayers`	Permite habilitar capas específicas de Astra Trident para el registro de seguimiento o la supresión de registros.	""
`tridentImage`	Permite anular por completo la imagen de Astra Trident.	""
`tridentImageTag`	Permite sobrescribir la etiqueta de la imagen para Astra Trident.	""
`tridentProbePort`	Permite sobrescribir el puerto predeterminado utilizado para las sondas de vida/preparación de Kubernetes.	""
`windows`	Permite instalar Astra Trident en el nodo de trabajo de Windows.	`false`
`enableForceDetach`	Permite habilitar la función Forzar separación.	`false`
`excludePodSecurityPolicy`	Excluye la política de seguridad del pod del operador de la creación.	`false`

Descripción de los pods de la controladora y los pods de nodo

Astra Trident se ejecuta como un único pod de la controladora, más un pod de nodos en cada nodo de trabajo del clúster. El pod del nodo debe ejecutarse en cualquier host en el que desee montar potencialmente un volumen Astra Trident.

Kubernetes "selectores de nodos" y.. "toleraciones y tintes" se utilizan para restringir un pod para ejecutarse en un nodo concreto o preferido. Uso del "ControllerPlugin" y. `NodePlugin`, puede especificar restricciones y anulaciones.

- El complemento de la controladora se ocupa del aprovisionamiento y la gestión de volúmenes, como snapshots y redimensionamiento.
- El complemento de nodo se encarga de conectar el almacenamiento al nodo.

El futuro

Ahora es posible "cree una clase de back-end y almacenamiento, aprovisione un volumen y monte el volumen en un pod".

Personalice la instalación del operador de Trident

El operador Trident le permite personalizar la instalación de Astra Trident con los atributos del `TridentOrchestrator` espec. Si desea personalizar la instalación más allá de qué `TridentOrchestrator` los argumentos lo permiten, considere usar `tridentctl` Para generar manifiestos YAML personalizados y modificarlos según sea necesario.

Descripción de los pods de la controladora y los pods de nodo

Astra Trident se ejecuta como un único pod de la controladora, más un pod de nodos en cada nodo de trabajo del clúster. El pod del nodo debe ejecutarse en cualquier host en el que desee montar potencialmente un volumen Astra Trident.

Kubernetes "selectores de nodos" y.. "toleraciones y tintes" se utilizan para restringir un pod para ejecutarse en un nodo concreto o preferido. Uso del "ControllerPlugin" y. `NodePlugin`, puede especificar restricciones y anulaciones.

- El complemento de la controladora se ocupa del aprovisionamiento y la gestión de volúmenes, como snapshots y redimensionamiento.
- El complemento de nodo se encarga de conectar el almacenamiento al nodo.

Opciones de configuración

`spec.namespace` se especifica en `TridentOrchestrator` Para indicar el espacio de nombres en el que está instalado Astra Trident. Este parámetro **no se puede actualizar después de instalar Astra Trident**. Al intentar hacerlo, se genera el `TridentOrchestrator` estado a cambiar a. `Failed`. Astra Trident no está pensado para la migración entre espacios de nombres.

Esta tabla detalla `TridentOrchestrator` atributos.

Parámetro	Descripción	Predeterminado
`namespace`	Espacio de nombres para instalar Astra Trident en	"predeterminado"
`debug`	Habilite la depuración para Astra Trident	falso
`windows`	Ajuste a. `true` Permite la instalación en nodos de trabajo de Windows.	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

Parámetro	Descripción	Predeterminado
`autosupportImage`	La imagen contenedora para telemetría AutoSupport	"netapp/trident-autosupport:23.01"
`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 1.19+) 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	
`imagePullSecrets`	Secretos para extraer imágenes de un registro interno	
`imagePullPolicy`	Establece la política de extracción de imágenes para el operador Trident. Valores válidos: `Always` para tirar siempre de la imagen. `IfNotPresent` para extraer la imagen solo si aún no existe en el nodo. `Never` para no tirar nunca de la imagen.	`IfNotPresent`
`controllerPluginNodeSelector`	Selectores de nodos adicionales para POD. Sigue el mismo formato que pod.spec.nodeSelector.	Sin valores predeterminados; opcional
`controllerPluginTolerations`	Anula la toleración de Kubernetes en pods. Sigue el mismo formato que el de pod.spec.tolerancias.	Sin valores predeterminados; opcional
`nodePluginNodeSelector`	Selectores de nodos adicionales para POD. Sigue el mismo formato que pod.spec.nodeSelector.	Sin valores predeterminados; opcional

Parámetro	Descripción	Predeterminado
`nodePluginTolerations`	Anula la toleración de Kubernetes en pods. Sigue el mismo formato que el de pod.spec.tolerancias.	Sin valores predeterminados; opcional

Para obtener más información sobre el formato de los parámetros del POD, consulte "Asignación de pods a nodos".

Configuraciones de ejemplo

Puede utilizar los atributos mencionados anteriormente al definir `TridentOrchestrator` para personalizar la instalación.

Ejemplo 1: Configuración personalizada básica

Este es un ejemplo de una configuración personalizada básica.

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

Ejemplo 2: Implementar con selectores de nodos

Este ejemplo ilustra cómo se puede implementar Trident con los selectores de nodos:

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

Ejemplo 3: Implementar en nodos de trabajo de Windows

Este ejemplo ilustra la implementación en un nodo de trabajo de Windows.

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

Instale utilizando trimentctl

Instale utilizando trimentctl

Puede instalar Astra Trident con `tridentctl`. Este proceso se aplica a instalaciones en las que las imágenes de contenedor requeridas por Astra Trident se almacenan o no en un registro privado. Para personalizar su `tridentctl` despliegue, consulte "Personalice la implementación trimentctl".

Información crucial sobre Astra Trident 23.01

- Debe leer la siguiente información crítica sobre Astra Trident.*

información bï¿½tico sobre Astra Tridbï¿½tico

- Kubernetes 1.26 ahora es compatible con Trident. Actualizar Trident antes de actualizar Kubernetes.
- Astra Trident cumple estrictamente el uso de la configuración de múltiples rutas en entornos SAN, con un valor recomendado de `find_multipaths: no` en el archivo multipath.conf.

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

Instale Astra Trident con `tridentctl`

Revisar "descripción general de la instalación" para asegurarse de cumplir con los requisitos previos de instalación y seleccionar la opción de instalación correcta para el entorno.

Antes de empezar

Antes de iniciar la instalación, inicie sesión en el host Linux y compruebe que esté gestionando un trabajo, "Clúster de Kubernetes compatible" y que tenga los privilegios necesarios.

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`.

1. Compruebe su versión de Kubernetes:

```
kubectl version
```

2. Comprobar los privilegios de administrador de clúster:

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

3. Compruebe que puede iniciar un pod que utilice una imagen de Docker Hub para llegar al sistema de almacenamiento a través de la red de pod:

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

Paso 1: Descargue el paquete de instalación de Trident

El paquete de instalación de Astra Trident crea un pod Trident, configura los objetos CRD que se utilizan para mantener su estado e inicializa las sidecs CSI para realizar acciones como aprovisionar y adjuntar volúmenes a los hosts del clúster. Descargue y extraiga la versión más reciente del instalador de Trident "La sección *Assets* de GitHub". Actualice *<trident-installer-XX.XX.X.tar.gz>* en el ejemplo con la versión Astra Trident seleccionada.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Paso 2: Instale Astra Trident

Instale Astra Trident en el espacio de nombres deseado ejecutando `tridentctl install` comando. Puede agregar argumentos adicionales para especificar la ubicación del registro de imágenes.

Para habilitar Astra Trident para que se ejecute en los nodos de Windows, añada `--windows` marque el comando install: `$ ./tridentctl install --windows -n trident`.

Modo estándar

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

Imágenes en un registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.01 --trident
-image <your-registry>/trident:23.01.1
```

Imágenes en diferentes registros

Debe añadir `sig-storage` para la `imageRegistry` para usar diferentes ubicaciones de registro.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.01 --trident-image <your-
registry>/netapp/trident:23.01.1
```

El estado de su instalación debería tener un aspecto parecido a este.

```
....
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=23.01.1
INFO Trident installation succeeded.
....
```

Compruebe la instalación

Puede verificar la instalación con el estado de creación de un pod o. `tridentctl`.

Uso del estado de creación de pod

Para confirmar si la instalación de Astra Trident ha finalizado, revise 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 **en ejecución**, la plataforma no estaba instalada. Uso `-d` para "activa el modo de depuración" y solucionar el problema.

Uso `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

El futuro

Ahora es posible "cree una clase de back-end y almacenamiento, aprovisione un volumen y monte el volumen en un pod".

Personalice la instalación trimentctl

Puede utilizar el instalador de Astra Trident para personalizar la instalación.

Obtenga más información sobre el instalador

El instalador de Astra 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]`.

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`
- `trident-resourcequota.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 instalar 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."