Realice puestas en marcha con el operador de Trident

Astra Trident

NetApp
September 04, 2024

Tabla de contenidos

Realice puestas en marcha con el operador de Trident

Puede poner en marcha Astra Trident con el operador de Trident.

Información crucial sobre Astra Trident 22.10

Debe leer la siguiente información crítica antes de actualizar a Astra Trident 22.10.

** información de las Ocampo sobre la Astra Trident 22.10 **

- Kubernetes 1.25 ahora es compatible con Trident. Debe actualizar a Astra Trident 22.10 antes de actualizar a Kubernetes 1.25.
- Astra Trident ahora cumple estrictamente el uso de la configuración de varias 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.

Opciones de implementación del operador Trident

El operador de Trident se puede poner en marcha de dos maneras:

- Usar Trident "Carta del timón": El gráfico Helm despliega el operador Trident e instala Trident en un solo paso.
- Manualmente: Trident proporciona un archivo que se puede usar para instalar el operador y crear objetos asociados.
  - 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".

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

Verifique los requisitos previos

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

- Tiene privilegios completos para un clúster de Kubernetes compatible que ejecute una versión de Kubernetes compatible. Revise la "requisitos".
- 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 e instale Astra Trident con 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:

- A. "Compatible con la versión de Kubernetes"
- Versión timón 3

Pasos

1. Añada el repositorio Helm de Trident:

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

2. Utilice la `helm install` y especifique un nombre para la implementación. Consulte el siguiente ejemplo:

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--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.

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> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace --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 e instale 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 verificar 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>
```

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 y extraiga la versión más reciente del paquete de instalación de Trident "La sección *Assets* de GitHub".

```
wget
https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.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:22.10
    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: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 instalación de Astra Trident con los atributos del `TridentOrchestrator` espec.

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

`spec.namespace` se especifica en `TridentOrchestrator` Para indicar qué espacio de nombres 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.

Opciones de configuración

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
`autosupportImage`	La imagen contenedora para telemetría AutoSupport	"netapp/trident-autosupport:22.10.0"
`autosupportProxy`	La dirección/puerto de un proxy para enviar telemetría AutoSupport	"http://proxy.example.com:8888"

Parámetro	Descripción	Predeterminado
`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	
`controllerPluginNodeSelector`	Selectores de nodos adicionales para POD que ejecutan el complemento Trident Controller CSI. Sigue el mismo formato que pod.spec.nodeSelector.	Sin valores predeterminados; opcional
`controllerPluginTolerations`	Anula la tolerancia de los pods que ejecutan el complemento CSI del controlador Trident. Sigue el mismo formato que el de pod.spec.tolerancias.	Sin valores predeterminados; opcional
`nodePluginNodeSelector`	Selectores de nodos adicionales para POD que ejecutan el complemento Trident Node CSI. Sigue el mismo formato que pod.spec.nodeSelector.	Sin valores predeterminados; opcional
`nodePluginTolerations`	Anula las toleraciones para los pods que ejecutan el complemento CSI de nodos Trident. 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
```