Realizar operaciones de volumen

Astra Trident

NetApp
November 20, 2023

Tabla de contenidos

Realizar operaciones de volumen

Más información sobre las funciones que ofrece Astra Trident para la gestión de volúmenes.

- "Utilice Topología CSI"
- "Trabajar con instantáneas"
- "Expanda los volúmenes"
- "Importar volúmenes"

Utilice Topología CSI

Astra Trident puede crear y conectar volúmenes a los nodos presentes en un clúster de Kubernetes de forma selectiva mediante el uso de "Función de topología CSI". Con la función de topología CSI, el acceso a los volúmenes puede limitarse a un subconjunto de nodos, en función de regiones y zonas de disponibilidad. En la actualidad, los proveedores de cloud permiten a los administradores de Kubernetes generar nodos basados en zonas. Los nodos pueden estar ubicados en distintas regiones de una zona de disponibilidad o en varias zonas de disponibilidad. Para facilitar el aprovisionamiento de volúmenes para cargas de trabajo en una arquitectura de varias zonas, Astra Trident utiliza la topología CSI.

> Obtenga más información sobre la característica de topología CSI "aquí".

Kubernetes ofrece dos modos de enlace de volúmenes únicos:

- Con `VolumeBindingMode` establezca en `Immediate`, Astra Trident crea el volumen sin conocimiento de la topología. La vinculación de volúmenes y el aprovisionamiento dinámico se manejan cuando se crea la RVP. Este es el valor predeterminado `VolumeBindingMode` y es adecuado para clústeres que no aplican restricciones de topología. Los volúmenes persistentes se crean sin dependencia alguna de los requisitos de programación del POD solicitante.
- Con `VolumeBindingMode` establezca en `WaitForFirstConsumer`, La creación y enlace de un volumen persistente para una RVP se retrasa hasta que se programa y crea un pod que usa la RVP. De esta forma, se crean volúmenes con el fin de cumplir las restricciones de programación que se aplican en los requisitos de topología.

> La `WaitForFirstConsumer` el modo de encuadernación no requiere etiquetas de topología. Esto se puede utilizar independientemente de la característica de topología CSI.

Lo que necesitará

Para utilizar la topología CSI, necesita lo siguiente:

- Un clúster de Kubernetes que funciona con la versión 1.17 o posterior.

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Los nodos del clúster deben tener etiquetas que incluyan el reconocimiento de topología (`topology.kubernetes.io/region` y.. `topology.kubernetes.io/zone`). Estas etiquetas * deben estar presentes en los nodos del clúster* antes de instalar Astra Trident para que Astra Trident tenga en cuenta la topología.

```
$ kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Paso 1: Cree un backend con detección de topología

Los back-ends de almacenamiento de Astra Trident se pueden diseñar para aprovisionar de forma selectiva volúmenes en función de las zonas de disponibilidad. Cada back-end puede llevar un opcional `supportedTopologies` bloque que representa una lista de zonas y regiones que se deben admitir. En el caso de StorageClasses que utilizan dicho back-end, solo se creará un volumen si lo solicita una aplicación programada en una región/zona admitida.

Este es el aspecto de una definición de backend de ejemplo:

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "xxxxxxxxxxxx",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

> `supportedTopologies` se utiliza para proporcionar una lista de regiones y zonas por backend. Estas regiones y zonas representan la lista de valores permitidos que se pueden proporcionar en un StorageClass. En el caso de StorageClasses que contienen un subconjunto de las regiones y zonas proporcionadas en un back-end, Astra Trident creará un volumen en el back-end.

Puede definir `supportedTopologies` por pool de almacenamiento también. Consulte el siguiente ejemplo:

```
{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"},
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
"storage": [
   {
       "labels": {"workload":"production"},
        "region": "Iowa-DC",
        "zone": "Iowa-DC-A",
        "supportedTopologies": [
            {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
        ]
    },
    {
        "labels": {"workload":"dev"},
         "region": "Iowa-DC",
         "zone": "Iowa-DC-B",
         "supportedTopologies": [
             {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
         ]
     }
]
}
```

En este ejemplo, la `region` y.. `zone` las etiquetas indican la ubicación del pool de almacenamiento. `topology.kubernetes.io/region` y.. `topology.kubernetes.io/zone` dicte desde donde se pueden consumir los pools de almacenamiento.

Paso 2: Defina las clases de almacenamiento que tienen en cuenta la topología

En función de las etiquetas de topología que se proporcionan a los nodos del clúster, se puede definir StorageClasse para que contenga información de topología. Esto determinará los pools de almacenamiento que sirven como candidatos para las solicitudes de RVP y el subconjunto de nodos que pueden usar los volúmenes aprovisionados mediante Trident.

Consulte el siguiente ejemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

En la definición del tipo de almacenamiento que se proporciona anteriormente, `volumeBindingMode` se establece en `WaitForFirstConsumer`. Las RVP solicitadas con este tipo de almacenamiento no se verán en cuestión hasta que se mencionan en un pod. Y, `allowedTopologies` proporciona las zonas y la región que se van a utilizar. La `netapp-san-us-east1` StorageClass creará EVs en el `san-backend-us-east1` backend definido anteriormente.

Paso 3: Cree y utilice un PVC

Con el clase de almacenamiento creado y asignado a un back-end, ahora puede crear RVP.

Vea el ejemplo `spec` a continuación:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

La creación de una RVP con este manifiesto daría como resultado lo siguiente:

```
$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
$ kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para que Trident cree un volumen y lo enlace a la RVP, use la RVP en un pod. Consulte el siguiente ejemplo:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Este podSpec indica a Kubernetes que programe el pod de los nodos presentes en el `us-east1` region y elija de cualquier nodo que esté presente en el `us-east1-a` o. `us-east1-b` zonas.

Consulte la siguiente salida:

```
$ kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
$ kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Actualice los back-ends que se incluirán `supportedTopologies`

Se pueden actualizar los back-ends preexistentes para incluir una lista de `supportedTopologies` uso `tridentctl backend update`. Esto no afectará a los volúmenes que ya se han aprovisionado, y sólo se utilizarán en las siguientes CVP.

Obtenga más información

- "Gestione recursos para contenedores"
- "Selector de nodos"
- "Afinidad y anti-afinidad"
- "Tolerancias y taints"

Trabajar con instantáneas

A partir del lanzamiento de Astra Trident 20.01, se pueden crear snapshots de VP en la capa de Kubernetes. Puede utilizar estas snapshots para mantener copias de un momento específico de los volúmenes que haya creado Astra Trident y programar la creación de volúmenes adicionales (clones). La copia de Snapshot de volumen es compatible con `ontap-nas`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `aws-cvs`, `gcp-cvs`, y. `azure-netapp-files` de windows

> Esta función está disponible en Kubernetes 1.17 (beta) y es GA a partir de 1.20. Para comprender los cambios relacionados con el paso de beta a GA, consulte "el blog del lanzamiento". Con la graduación a GA, el `v1` Se introduce la versión API y es compatible con versiones anteriores `v1beta1` snapshot.

Lo que necesitará

- Para crear snapshots de volumen, es necesario crear una controladora de snapshot externa, así como algunas definiciones de recursos personalizados (CRD). Esta es la responsabilidad del orquestador de Kubernetes que se está utilizando (por ejemplo: Kubeadm, GKE, OpenShift).

Es posible crear una controladora Snapshot externa y CRD Snapshot de la siguiente manera:

1. Crear CRD de snapshot de volumen:

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crear la controladora Snapshot en el espacio de nombres que desee. Edite los manifiestos YAML a continuación para modificar el espacio de nombres.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

CSI Snapshotter proporciona un "validando webhook" ayudar a los usuarios a validar las instantáneas v1beta1 existentes y confirmar que son objetos de recursos válidos. El gancho web validador etiqueta automáticamente objetos de instantánea no válidos e impide la creación de futuros objetos no válidos. Kubernetes orchestrator pone en marcha el enlace validando webhook. Consulte las instrucciones para implementar manualmente el enlace web validador "aquí". Busque ejemplos de manifiestos de instantáneas no válidos "aquí".

El ejemplo detallado a continuación explica las construcciones necesarias para trabajar con instantáneas y muestra cómo se pueden crear y usar las instantáneas.

Paso 1: Configurar un `VolumeSnapshotClass`

Antes de crear una snapshot de volumen, establezca un enlace:./trident-reference/objects.html[`VolumeSnapshotClass`#].

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

La `driver` Apunta al driver CSI de Astra Trident. `deletionPolicy` puede ser `Delete` o. `Retain`. Cuando se establece en `Retain`, la instantánea física subyacente en el clúster de almacenamiento se conserva incluso cuando `VolumeSnapshot` el objeto se ha eliminado.

Paso 2: Crear una instantánea de una RVP existente

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

La copia de Snapshot se está creando para una RVP llamada `pvc1`, y el nombre de la instantánea se establece en `pvc1-snap`.

```
$ kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

$ kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

Esto creó un `VolumeSnapshot` objeto. Un VolumeSnapshot es análogo a un PVC y está asociado a un `VolumeSnapshotContent` objeto que representa la instantánea real.

Es posible identificar la `VolumeSnapshotContent` objeto para `pvc1-snap` VolumeSnapshot, describiéndolo.

```
$ kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

La `Snapshot Content Name` Identifica el objeto VolumeSnapshotContent que sirve esta snapshot. La `Ready To Use` Parámetro indica que la Snapshot se puede usar para crear una RVP nueva.

Paso 3: Creación de EVs a partir de VolumeSnapshots

Consulte el siguiente ejemplo para crear una RVP con una Snapshot:

```
$ cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

`dataSource` Muestra que la RVP debe crearse con un VolumeSnapshot llamado `pvc1-snap` como la fuente de los datos. Esto le indica a Astra Trident que cree una RVP a partir de la snapshot. Una vez creada la RVP, se puede conectar a un pod y utilizarla como cualquier otro PVC.

> Cuando se elimina un volumen persistente con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un “estado de eliminación”. Para eliminar el volumen Astra Trident, deben eliminarse las snapshots del volumen.

Obtenga más información

- "Copias de Snapshot de volumen"
- enlace:../trident-reference/objects.html[`VolumeSnapshotClass`#]

Expanda los volúmenes

Astra Trident ofrece a los usuarios de Kubernetes la capacidad de ampliar sus volúmenes una vez que se han creado. Encuentre información sobre las configuraciones que se necesitan para ampliar los volúmenes iSCSI y NFS.

Expanda un volumen iSCSI

Puede expandir un volumen persistente iSCSI (PV) mediante el aprovisionador CSI.

> La ampliación del volumen iSCSI se admite en el `ontap-san`, `ontap-san-economy`, `solidfire-san` Requiere Kubernetes 1.16 o posterior.

Descripción general

Para expandir un VP iSCSI, se deben realizar los siguientes pasos:

- Editar la definición de StorageClass para establecer el `allowVolumeExpansion` campo a. `true`.
- Edición de la definición de PVC y actualización de `spec.resources.requests.storage` para reflejar el nuevo tamaño deseado, que debe ser mayor que el tamaño original.
- Para que se pueda cambiar el tamaño, se debe conectar el PV a un pod. Existen dos situaciones a la hora de cambiar el tamaño de un VP iSCSI:
  - Si el VP está conectado a un pod, Astra Trident amplía el volumen en el back-end de almacenamiento, vuelve a buscar el dispositivo y cambia el tamaño del sistema de archivos.
  - Cuando se intenta cambiar el tamaño de un VP sin conectar, Astra Trident amplía el volumen en el back-end de almacenamiento. Una vez que la RVP está Unido a un pod, Trident vuelve a buscar el dispositivo y cambia el tamaño del sistema de archivos. Kubernetes, después, actualiza el tamaño de RVP después de completar correctamente la operación de ampliación.

El ejemplo siguiente muestra cómo funcionan las VP iSCSI.

Paso 1: Configure el tipo de almacenamiento para que admita la ampliación de volumen

```
$ cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

En el caso de un tipo de almacenamiento existente, edítelo para incluir el `allowVolumeExpansion` parámetro.

Paso 2: Cree una RVP con el tipo de almacenamiento que ha creado

```
$ cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident crea un volumen persistente (PV) y lo asocia con esta solicitud de volumen persistente (PVC).

```
$ kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               10s
```

Paso 3: Defina un pod que fije el PVC

En este ejemplo, se crea un pod que utiliza `san-pvc`.

```
$  kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
centos-pod   1/1     Running   0          65s

$  kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    centos-pod
```

Paso 4: Expanda el PV

Para cambiar el tamaño del VP que se ha creado de 1GI a 2gi, edite la definición de PVC y actualice el `spec.resources.requests.storage` A 2gi.

```
$ kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

Paso 5: Validar la expansión

Para validar que la ampliación ha funcionado correctamente, compruebe el tamaño del volumen PVC, PV y Astra Trident:

```
$ kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
$ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
$ tridentctl get volumes -n trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Expanda un volumen NFS

Astra Trident admite la ampliación de volúmenes para los VP de NFS aprovisionados en `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `aws-cvs`, `gcp-cvs`, y. `azure-netapp-files` back-ends.

Paso 1: Configure el tipo de almacenamiento para que admita la ampliación de volumen

Para cambiar el tamaño de un VP de NFS, el administrador primero tiene que configurar la clase de almacenamiento para permitir la expansión del volumen estableciendo el `allowVolumeExpansion` campo a. `true`:

```
$ cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si ya ha creado una clase de almacenamiento sin esta opción, puede simplemente editar la clase de almacenamiento existente mediante `kubectl edit storageclass` para permitir la expansión de volumen.

Paso 2: Cree una RVP con el tipo de almacenamiento que ha creado

```
$ cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident debe crear un PV NFS de 20 MiB para esta RVP:

```
$ kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Paso 3: Expanda el PV

Para cambiar el tamaño del VP de 20 MiB recién creado a 1 GIB, edite el RVP y establezca `spec.resources.requests.storage` A 1 GB:

```
$ kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

Paso 4: Validar la expansión

Puede validar que el tamaño de la configuración ha funcionado correctamente comprobando el tamaño del volumen PVC, PV y Astra Trident:

```
$ kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

$ tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n
trident
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Importar volúmenes

Es posible importar volúmenes de almacenamiento existentes como un VP de Kubernetes mediante `tridentctl import`.

Controladores que admiten la importación de volúmenes

En esta tabla se muestran los controladores que admiten la importación de volúmenes y la versión en la que se introdujeron.

Controlador	Liberar
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04

Controlador	Liberar
`aws-cvs`	19.04
`azure-netapp-files`	19.04
`gcp-cvs`	19.04
`ontap-san`	19.04

¿Por qué debo importar volúmenes?

Existen varios casos de uso para importar un volumen en Trident:

- Contenerización de una aplicación y reutilización del conjunto de datos existente
- Usar un clon de un conjunto de datos para una aplicación efímera
- Reconstruir un clúster de Kubernetes con fallos
- Migración de datos de aplicaciones durante la recuperación tras siniestros

¿Cómo funciona la importación?

El proceso de importación de volúmenes utiliza el archivo de solicitud de volumen persistente (PVC) para crear la RVP. Como mínimo, el archivo PVC debe incluir los campos name, Namespace, accessModes y storageClassName como se muestra en el ejemplo siguiente.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

La `tridentctl` el cliente se utiliza para importar un volumen de almacenamiento existente. Trident importa el volumen persiste en los metadatos del volumen y crea la RVP y el VP.

```
$ tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

Para importar un volumen de almacenamiento, especifique el nombre del back-end de Astra Trident que contiene el volumen, y el nombre que identifica de forma única el volumen en el almacenamiento (por ejemplo: ONTAP FlexVol, Element Volume, CVS Volume path). El volumen de almacenamiento debe permitir el acceso de lectura/escritura y debe ser accesible desde el back-end de Astra Trident especificado. La `-f` El argumento

String es necesario y especifica la ruta al archivo YLMA o PVC JSON.

Cuando Astra Trident recibe la solicitud de importación de volumen, se determina el tamaño de volumen existente y se establece en la RVP. Una vez que el controlador de almacenamiento importa el volumen, se crea el PV con un ClaimRef al PVC. La política de reclamaciones se establece inicialmente en `retain` En el PV. Una vez que Kubernetes enlaza correctamente la RVP y el VP, se actualiza la política de reclamaciones para que coincida con la política de reclamaciones de la clase de almacenamiento. Si la política de reclamaciones de la clase de almacenamiento es `delete`, El volumen de almacenamiento se eliminará cuando se elimine el PV.

Cuando se importa un volumen con `--no-manage` Argumento, Trident no realiza ninguna operación adicional en la RVP o el VP durante el ciclo de vida de los objetos. Dado que Trident ignora los eventos VP y RVP para `--no-manage` Los objetos, el volumen de almacenamiento no se elimina cuando se elimina el VP. También se ignoran otras operaciones como el clon de volumen y el cambio de tamaño de volumen. Esta opción es útil si desea usar Kubernetes para cargas de trabajo en contenedores, pero de lo contrario desea gestionar el ciclo de vida del volumen de almacenamiento fuera de Kubernetes.

Se agrega una anotación a la RVP y al VP que tiene el doble propósito de indicar que el volumen se importó y si se administran la PVC y la VP. Esta anotación no debe modificarse ni eliminarse.

Trident 19.07 y versiones posteriores gestionan el adjunto de los VP y monta el volumen como parte de la importación. Para las importaciones con versiones anteriores de Astra Trident, no habrá ninguna operación en la ruta de datos y la importación de volúmenes no verificará si es posible montar el volumen. Si se produce un error con la importación de volumen (por ejemplo, StorageClass es incorrecto), puede recuperar cambiando la política de reclamación en el VP a. `retain`, Eliminando el PVC y el VP y volviendo a intentar el comando de importación de volumen.

`ontap-nas` y.. `ontap-nas-flexgroup` importaciones

Cada volumen creado con `ontap-nas` Driver es una FlexVol en el clúster de ONTAP. Importación de FlexVols con `ontap-nas` el controlador funciona igual. Una FlexVol que ya existe en un clúster de ONTAP se puede importar como `ontap-nas` RVP. Del mismo modo, los volúmenes FlexGroup se pueden importar del mismo modo `ontap-nas-flexgroup` EVs.

> Un volumen de ONTAP debe ser del tipo rw que haya que importar Trident. Si un volumen es del tipo dp, es un volumen de destino de SnapMirror, se debe interrumpir la relación de mirroring antes de importar el volumen a Trident.

> La `ontap-nas` el controlador no puede importar y gestionar qtrees. La `ontap-nas` y.. `ontap-nas-flexgroup` las controladoras no permiten nombres de volúmenes duplicados.

Por ejemplo, para importar un volumen llamado `managed_volume` en un backend llamado `ontap_nas`, utilice el siguiente comando:

```
$ tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Para importar un volumen denominado `unmanaged_volume` (en la `ontap_nas backend`), que Trident no administrará, utilice el siguiente comando:

```
$ tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

Cuando utilice la `--no-manage` Argumento, Trident no cambia el nombre del volumen ni se valida si se montó el volumen. Se produce un error en la operación de importación de volumen si el volumen no se montó manualmente.

> Se ha solucionado un error existente con la importación de volúmenes con UnixPermissions personalizado. Puede especificar unixPermissions en la definición de PVC o en la configuración de back-end, e indicar a Astra Trident que importe el volumen según corresponda.

`ontap-san` importar

Astra Trident también puede importar SAN FlexVols de ONTAP que contienen una única LUN. Esto es consistente con `ontap-san` Controlador, que crea una FlexVol para cada RVP y una LUN dentro del FlexVol. Puede utilizar el `tridentctl import` comando de la misma forma que en otros casos:

- Incluya el nombre del `ontap-san` back-end.

- Escriba el nombre de la FlexVol que se debe importar. Recuerde, esta FlexVol solo contiene una LUN que es necesario importar.
- Proporcione la ruta de la definición de PVC que debe utilizarse con el `-f` bandera.
- Elija entre administrar o no administrar el PVC. De forma predeterminada, Trident gestionará la RVP y cambiará el nombre de los FlexVol y LUN en el back-end. Para importar como volumen no administrado, pase el `--no-manage` bandera.

> Al importar un no administrado `ontap-san` Volumen, debe asegurarse de que el nombre de la LUN de la FlexVol sea `lun0` y se asigna a un igroup con los iniciadores deseados. Astra Trident se encarga automáticamente de esto en una importación gestionada.

A continuación, Astra Trident importará el FlexVol y lo asociará con la definición de PVC. Astra Trident también cambia el nombre de FlexVol al `pvc-<uuid>` Formatear y la LUN dentro de la FlexVol a. `lun0`.

> Se recomienda importar volúmenes que no tengan conexiones activas existentes. Si desea importar un volumen que está utilizado activamente, Clone el volumen primero y, a continuación, realice la importación.

Ejemplo

Para importar la `ontap-san-managed` FlexVol que está presente en el `ontap_san_default` back-end, ejecute el `tridentctl import` comando como:

```
$ tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

> Un volumen ONTAP debe ser del tipo rw que importe Astra Trident. Si un volumen es del tipo dp, es un volumen de destino de SnapMirror, se debe interrumpir la relación de mirroring antes de importar el volumen a Astra Trident.

`element` importar

Es posible importar el software NetApp Element/volúmenes de HCI de NetApp en el clúster de Kubernetes con Trident. Necesita el nombre de su entorno de administración Astra Trident, y el nombre único del volumen y el archivo PVC como argumentos para `tridentctl import` comando.

```
$ tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

El controlador Element admite los nombres de volúmenes duplicados. Si hay nombres de volúmenes duplicados, el proceso de importación de volúmenes de Trident devuelve un error. Como solución alternativa, Clone el volumen y proporcione un nombre de volumen único. A continuación, importe el volumen clonado.

`aws-cvs` importar

Para importar un volumen respaldado por Cloud Volumes Service de NetApp en AWS, identifique el volumen por su ruta de volumen en lugar de su nombre.

Para importar una `aws-cvs` volumen en el backend llamado `awscvs_YEppr` con la ruta del volumen de `adroit-jolly-swift`, utilice el siguiente comando:

```
$ tridentctl import volume awscvs_YEppr adroit-jolly-swift -f <path-to-
pvc-file> -n trident

+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | aws-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+------------------------------------------+--------+---------------
+----------+--------------------------------------+--------+---------+
```

La ruta del volumen es la parte de la ruta de exportación del volumen después de :/. Por ejemplo, si la ruta de exportación es `10.0.0.1:/adroit-jolly-swift`, la ruta de volumen es `adroit-jolly-swift`.

`gcp-cvs` importar

Importar un `gcp-cvs` el volumen funciona igual que la importación de un `aws-cvs` volumen.

`azure-netapp-files` importar

Para importar una `azure-netapp-files` volumen en el backend llamado `azurenetappfiles_40517` con la ruta del volumen `importvol1`, ejecute el siguiente comando:

```
$ tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+------------------------------------------+---------+---------------
+----------+--------------------------------------+--------+---------+
```

> La ruta de volumen para el volumen ANF está presente en la ruta de montaje después de :/. Por ejemplo, si la ruta de montaje es `10.0.0.2:/importvol1`, la ruta de volumen es `importvol1`.