



Realizar operaciones de volumen

Astra Trident

NetApp
November 14, 2025

Tabla de contenidos

Realizar operaciones de volumen	1
Utilice Topología CSI	1
Descripción general	1
Paso 1: Cree un backend con detección de topología	2
Paso 2: Defina las clases de almacenamiento que tienen en cuenta la topología	4
Paso 3: Cree y utilice un PVC	5
Actualice los back-ends que se incluirán <code>supportedTopologies</code>	8
Obtenga más información	8
Trabajar con instantáneas	8
Paso 1: Cree un <code>VolumeSnapshotClass</code>	9
Paso 2: Crear una instantánea de una RVP existente	9
Paso 3: Creación de EVs a partir de <code>VolumeSnapshots</code>	10
Eliminación de un VP con instantáneas	11
Implementar una controladora Snapshot de volumen	11
Recuperar datos de volumen mediante copias Snapshot	12
Enlaces relacionados	13
Expanda los volúmenes	13
Expanda un volumen iSCSI	13
Expanda un volumen NFS	17
Importar volúmenes	20
Descripción general y consideraciones	20
Importe un volumen	21
Ejemplos	22

Realizar operaciones de volumen

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](#)".

Descripción general

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 se pueden ubicar en diferentes zonas de disponibilidad dentro de una región o en varias regiones. 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 ejecuta un "[Compatible con la versión de Kubernetes](#)"

```

kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
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:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
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},\n{.metadata.labels}}{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.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","kubernetes.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","kubernetes.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.

A continuación se muestra un ejemplo de definición de backend:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
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
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "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: password
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
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` usando `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

Es posible crear snapshots de Kubernetes (snapshot de volumen) de volúmenes persistentes (VP) para mantener copias de un momento específico de los volúmenes Astra Trident. Además, es posible crear un nuevo volumen, también conocido como *clone*, a partir de una snapshot de volumen existente. Admite copias de Snapshot de volumen `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, y `azure-netapp-files` de windows

Antes de empezar

Debe tener un controlador de instantánea externo y definiciones de recursos personalizados (CRD). Esta es la responsabilidad del orquestador de Kubernetes (por ejemplo: Kubeadm, GKE, OpenShift).

Si su distribución de Kubernetes no incluye el controlador de instantáneas ni los CRD, consulte [Implementar una controladora Snapshot de volumen](#).



No cree una controladora Snapshot si crea instantáneas de volumen bajo demanda en un entorno de GKE. GKE utiliza un controlador de instantáneas oculto integrado.

Paso 1: Cree un VolumeSnapshotClass

En este ejemplo, se crea una clase de snapshot de volumen.

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

La driver Señala el controlador 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.

Para obtener más información, consulte el enlace:[./trident-reference/objects.html#kubernetes-volumesnapshotclass-objects\[VolumeSnapshotClass\]](#).

Paso 2: Crear una instantánea de una RVP existente

En este ejemplo, se crea una copia Snapshot de una RVP existente.

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

En este ejemplo, la snapshot se crea para una RVP llamada pvc1 y el nombre de la copia de snapshot 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

En este ejemplo, se crea una RVP con una copia 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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

dataSource Muestra que la RVP debe crearse con un VolumeSnapshot llamado `pvcl-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.



La RVP debe crearse en el mismo espacio de nombres que su `dataSource`.

Eliminación de un VP con instantáneas

Cuando se elimina un volumen persistente con instantáneas asociadas, el volumen Trident correspondiente se actualiza a un “estado de eliminación”. Quite las snapshots de volumen para eliminar el volumen de Astra Trident.

Implementar una controladora Snapshot de volumen

Si su distribución de Kubernetes no incluye el controlador de snapshots y los CRD, puede implementarlos de la siguiente manera.

Pasos

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-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
1
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Cree la controladora Snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



Si es necesario, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-
controller.yaml` y actualícelo `namespace` en el espacio de nombres.

Recuperar datos de volumen mediante copias Snapshot

El directorio de snapshots está oculto de forma predeterminada para facilitar la máxima compatibilidad de los volúmenes aprovisionados con el `ontap-nas` y `ontap-nas-economy` de windows. Habilite el `.snapshot` directorio para recuperar datos de snapshots directamente.

Use la interfaz de línea de comandos de ONTAP para restaurar un volumen en un estado registrado en una snapshot anterior.

```
cluster1::>*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Cuando se restaura una copia Snapshot, se sobrescribe la configuración de volúmenes existente. Se pierden los cambios que se hagan en los datos del volumen después de crear la copia Snapshot.

Enlaces relacionados

- "[Copias de Snapshot de volumen](#)"
- "[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
ubuntu-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:    ubuntu-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`, `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: Expande 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`.

Descripción general y consideraciones

Es posible importar un volumen en Astra Trident para lo siguiente:

- Agrupe en contenedores una aplicación y vuelva a utilizar su conjunto de datos existente
- Utilice el clon de un conjunto de datos para una aplicación efímera
- Reconstruya un clúster de Kubernetes que haya fallado
- Migración de datos de aplicaciones durante la recuperación ante desastres

Consideraciones

Antes de importar un volumen, revise las siguientes consideraciones.

- Astra Trident solo puede importar volúmenes de ONTAP de tipo RW (lectura y escritura). Los volúmenes del tipo DP (protección de datos) son volúmenes de destino de SnapMirror. Debe romper la relación de

reflejo antes de importar el volumen a Astra Trident.

- Sugerimos importar volúmenes sin conexiones activas. Para importar un volumen que se usa activamente, clone el volumen y, a continuación, realice la importación.



Esto es especialmente importante en el caso de volúmenes de bloque, ya que Kubernetes no sabía que la conexión anterior y podría conectar fácilmente un volumen activo a un pod. Esto puede provocar daños en los datos.

- Sin embargo StorageClass Debe especificarse en una RVP, Astra Trident no utiliza este parámetro durante la importación. Durante la creación de volúmenes, se usan las clases de almacenamiento para seleccionar entre los pools disponibles según las características de almacenamiento. Como el volumen ya existe, no se requiere ninguna selección de pool durante la importación. Por lo tanto, la importación no fallará incluso si el volumen existe en un back-end o pool que no coincide con la clase de almacenamiento especificada en la RVP.
- El tamaño del volumen existente se determina 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.
- De forma predeterminada, Astra Trident gestiona la RVP y cambia el nombre de FlexVol y LUN en el back-end. Puede pasar el `--no-manage` indicador para importar un volumen no gestionado. Si utiliza `--no-manage`, Astra Trident no realiza ninguna operación adicional en el PVC o PV durante el ciclo de vida de los objetos. El volumen de almacenamiento no se elimina cuando se elimina el VP, y 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.

Importe un volumen

Puede utilizar `tridentctl import` para importar un volumen.

Pasos

1. Cree el archivo de reclamación de volumen persistente (RVP) (por ejemplo, `pvc.yaml`) Que se utilizará para crear la RVP. El archivo PVC debe incluir `name`, `namespace`, `accessModes`, y `storageClassName`. Opcionalmente, se puede especificar `unixPermissions` En su definición de PVC.

A continuación se muestra un ejemplo de una especificación mínima:

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



No incluya parámetros adicionales, como el nombre del VP o el tamaño del volumen. Esto puede provocar un error en el comando de importación.

2. Utilice la `tridentctl import` Comando para especificar 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, ruta Cloud Volumes Service). La `-f` Se necesita un argumento para especificar la ruta al archivo PVC.

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

Ejemplos

Revise los siguientes ejemplos de importación de volúmenes para los controladores compatibles.

NAS de ONTAP y NAS FlexGroup de ONTAP

Astra Trident admite la importación de volúmenes mediante el `ontap-nas` y.. `ontap-nas-flexgroup` de windows



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

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.

Ejemplos de NAS de ONTAP

A continuación, se muestra un ejemplo de un volumen gestionado y una importación de volumen no gestionada.

Volumen gestionado

En el ejemplo siguiente se importa un volumen llamado `managed_volume` en un backend llamado `ontap_nas`:

```
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     |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Volumen no gestionado

Cuando utilice la `--no-manage` Argumento, Astra Trident no cambia el nombre del volumen.

El siguiente ejemplo importa `unmanaged_volume` en la `ontap_nas` backend:

```
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     |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

SAN de ONTAP

Astra Trident admite la importación de volúmenes mediante el `ontap-san` controlador.

Astra Trident puede importar volúmenes FlexVol de SAN de ONTAP que contengan una única LUN. Esto es consistente con `ontap-san` Controlador, que crea una FlexVol para cada RVP y una LUN dentro del FlexVol. Astra Trident importa el FlexVol y lo asocia con la definición de RVP.

Ejemplos de SAN de ONTAP

A continuación, se muestra un ejemplo de un volumen gestionado y una importación de volumen no gestionada.

Volumen gestionado

En el caso de los volúmenes gestionados, Astra Trident cambia el nombre del FlexVol al pvc-<uuid>. Formatear y la LUN dentro de la FlexVol a. lun0.

El siguiente ejemplo importa el `ontap-san-managed` FlexVol que está presente en el `ontap_san_default` backend:

```
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       |
+-----+-----+-----+
+-----+-----+-----+
```

Volumen no gestionado

El siguiente ejemplo importa `unmanaged_example_volume` en la `ontap_san` backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage

+-----+-----+-----+
+-----+-----+-----+
|           NAME          |  SIZE   | STORAGE CLASS |
PROTOCOL |           BACKEND UUID        | STATE   | MANAGED    |
+-----+-----+-----+
+-----+-----+-----+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog     |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online  | false      |
+-----+-----+-----+
+-----+-----+-----+
```

Si tiene LUN asignadas a iGroups que comparten un IQN con un IQN de nodo de Kubernetes, como se muestra en el ejemplo siguiente, recibirá el error: `LUN already mapped to initiator(s) in this group`. Deberá quitar el iniciador o desasignar la LUN para importar el volumen.

Vserver	Igroup	Protocol	OS	Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0	
svm0	unmanaged-example-igroup	mixed	linux		iqn.1994-05.com.redhat:4c2e1cf35e0

Elemento

Astra Trident admite el software NetApp Element y la importación de volúmenes de NetApp HCI mediante el solidfire-san controlador.



El controlador Element admite los nombres de volúmenes duplicados. Sin embargo, Astra Trident devuelve un error si hay nombres de volúmenes duplicados. Como solución alternativa, clone el volumen, proporcione un nombre de volumen único e importe el volumen clonado.

Ejemplo de elemento

El siguiente ejemplo importa un element-managed volumen en el back-end element_default.

```
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       |
+-----+-----+-----+
+-----+-----+-----+
```

Google Cloud Platform

Astra Trident admite la importación de volúmenes mediante el gcp-cvs controlador.



Para importar un volumen respaldado por NetApp Cloud Volumes Service en Google Cloud Platform, identifique el volumen según la ruta de volumen. La ruta del volumen es la parte de la ruta de exportación del volumen después del :/. Por ejemplo, si la ruta de exportación es 10.0.0.1:/adroit-jolly-swift, la ruta de volumen es adroit-jolly-swift.

Ejemplo de Google Cloud Platform

El siguiente ejemplo importa a gcp-cvs volumen en el back-end gpcvsv_YEppr con la ruta del volumen de adroit-jolly-swift.

```
tridentctl import volume gpcvsv_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 | gcp-storage  | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true       |
+-----+-----+
+-----+-----+
```

Azure NetApp Files

Astra Trident admite la importación de volúmenes mediante el azure-netapp-files y.. azure-netapp-files-subvolume de windows

 Para importar un volumen de Azure NetApp Files, identifique el volumen por su ruta de volumen. La ruta del volumen es la parte de la ruta de exportación del volumen después del :/. Por ejemplo, si la ruta de montaje es 10.0.0.2:/importvol1, la ruta de volumen es importvol1.

Ejemplo de Azure NetApp Files

El siguiente ejemplo importa un azure-netapp-files volumen en el back-end azurenatappfiles_40517 con la ruta del volumen importvol1.

```
tridentctl import volume azurenatappfiles_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       |
+-----+-----+
+-----+-----+
```

Información de copyright

Copyright © 2025 NetApp, Inc. Todos los derechos reservados. Impreso en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

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

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

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

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

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

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

Información de la marca comercial

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