



# **Executar operações de volume**

## **Astra Trident**

NetApp  
January 31, 2025

# Índice

- Executar operações de volume ..... 1
  - Use a topologia CSI ..... 1
  - Trabalhar com instantâneos ..... 8
  - Expandir volumes ..... 13
  - Importar volumes ..... 20

# Executar operações de volume

Saiba mais sobre os recursos fornecidos pelo Astra Trident para gerenciar seus volumes.

- ["Use a topologia CSI"](#)
- ["Trabalhar com instantâneos"](#)
- ["Expanda volumes"](#)
- ["Importar volumes"](#)

## Use a topologia CSI

O Astra Trident pode criar e anexar volumes de forma seletiva a nós presentes em um cluster Kubernetes usando o ["Recurso de topologia CSI"](#). Usando o recurso de topologia de CSI, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Hoje em dia, os provedores de nuvem permitem que os administradores do Kubernetes gerem nós baseados em zonas. Os nós podem ser localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para workloads em uma arquitetura de várias zonas, o Astra Trident usa topologia de CSI.



Saiba mais sobre o recurso de topologia de CSI ["aqui"](#).

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- `VolumeBindingMode`Com o definido como `Immediate`, o Astra Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Esse é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Os volumes persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um volume persistente para um PVC é adiada até que um pod que usa o PVC seja programado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.



O `WaitForFirstConsumer` modo de encadernação não requer rótulos de topologia. Isso pode ser usado independentemente do recurso de topologia de CSI.

### O que você vai precisar

Para fazer uso da topologia de CSI, você precisa do seguinte:

- Um cluster do Kubernetes executando o 1,19 -1,24.

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
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:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Os nós no cluster devem ter rótulos que introduzam reconhecimento da topologia (topology.kubernetes.io/region`e `topology.kubernetes.io/zone). Esses rótulos **devem estar presentes nos nós no cluster** antes que o Astra Trident seja instalado para que o Astra Trident esteja ciente da topologia.

```
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","kuber-
netes.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","kuber-
netes.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","kuber-
netes.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"}]
```

## Etapa 1: Crie um back-end com reconhecimento de topologia

Os back-ends de storage do Astra Trident podem ser desenvolvidos para provisionar volumes de forma seletiva, com base nas zonas de disponibilidade. Cada back-end pode transportar um bloco opcional `supportedTopologies` que representa uma lista de zonas e regiões que devem ser suportadas. Para o `StorageClasses` que fazem uso de tal back-end, um volume só seria criado se solicitado por um aplicativo agendado em uma região/zona suportada.

Veja como é um exemplo de definição de backend:

```
{
  "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` é usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um `StorageClass`. Para os `StorageClasses` que contêm um subconjunto das regiões e zonas fornecidas em um back-end, o Astra Trident criará um volume no back-end.

Você também pode definir `supportedTopologies` por pool de armazenamento. Veja o exemplo a seguir:

```

{"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"}
    ]
  }
]
}

```

Neste exemplo, as `region` etiquetas e `zone` representam a localização do conjunto de armazenamento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` e `zone` define de onde os pools de storage podem ser consumidos.

## Etapa 2: Defina StorageClasses que estejam cientes da topologia

Com base nas etiquetas de topologia fornecidas aos nós no cluster, o StorageClasses pode ser definido para conter informações de topologia. Isso determinará os pools de storage que atuam como candidatos a solicitações de PVC feitas e o subconjunto de nós que podem fazer uso dos volumes provisionados pelo Trident.

Veja o exemplo a seguir:

```
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"
```

Na definição StorageClass fornecida acima, volumeBindingMode está definida como WaitForFirstConsumer. Os PVCs solicitados com este StorageClass não serão utilizados até que sejam referenciados em um pod. E, allowedTopologies fornece as zonas e a região a serem usadas. O netapp-san-us-east1 StorageClass criará PVCs no san-backend-us-east1 back-end definido acima.

### Passo 3: Criar e usar um PVC

Com o StorageClass criado e mapeado para um back-end, agora você pode criar PVCs.

Veja o exemplo spec abaixo:

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

Criar um PVC usando este manifesto resultaria no seguinte:

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 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
  ----          -
  Normal        WaitForFirstConsumer 6s    persistentvolume-controller
waiting
for first consumer to be created before binding

```

Para o Trident criar um volume e vinculá-lo ao PVC, use o PVC em um pod. Veja o exemplo a seguir:



```

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: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

Este podSpec instrui o Kubernetes a agendar o pod em nós presentes na us-east1 região e escolher entre qualquer nó presente nas us-east1-a zonas ou us-east1-b.

Veja a seguinte saída:

```
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
```

## Atualize os backends para incluir `supportedTopologies`

Os backends pré-existentes podem ser atualizados para incluir uma lista `supportedTopologies` de uso ``tridentctl backend update`` do . Isso não afetará os volumes que já foram provisionados e só será usado para PVCs subsequentes.

## Encontre mais informações

- ["Gerenciar recursos para contêineres"](#)
- ["NodeSelector"](#)
- ["Afinidade e anti-afinidade"](#)
- ["Taints e Tolerations"](#)

## Trabalhar com instantâneos

A partir do lançamento de 20,01 do Astra Trident, você pode criar snapshots de PVS na camada Kubernetes. Use esses snapshots para manter cópias pontuais de volumes criados pelo Astra Trident e agendar a criação de volumes adicionais (clones). O instantâneo de volume é suportado pelos `ontap-nas` drivers , `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs` e `azure-netapp-files` .



Esse recurso está disponível no Kubernetes 1,17 (beta) e é GA no 1,20. Para entender as mudanças envolvidas na mudança de beta para GA, ["o blog de lançamento"](#) consulte . Com a graduação para GA, a v1 versão da API é introduzida e é compatível com v1beta1 snapshots.

### O que você vai precisar

- A criação de instantâneos de volume requer a criação de um controlador de snapshot externo e de definições personalizadas de recursos (CRDs). Essa é a responsabilidade do orquestrador do Kubernetes sendo usado (por exemplo: Kubeadm, GKE, OpenShift).

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

1. Criar CRDs de instantâneos de volume.

Para Kubernetes 1,20 e posterior, use CRDs de snapshot de v1 com componentes de snapshot de v5,0 ou superior. Para Kubernetes 1,19, use o v1beta1 com componentes de snapshot v3,0.3.

## v5.0 componentes

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

## v3.0.3 componentes

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshots.yaml
```

2. Crie o controlador de snapshot no namespace desejado. Edite os manifestos YAML abaixo para modificar o namespace.

Para Kubernetes 1,20 e posterior, use o v5,0 ou superior. Para a versão 1,19 do Kubernetes, use a v3,0.3



Não crie um controlador de instantâneos se configurar instantâneos de volume sob demanda em um ambiente GKE. O GKE utiliza um controlador instantâneo oculto incorporado.

### Controlador v5.0

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

### Controlador v3.0.3

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```



O CSI Snapshotter fornece um "validar webhook" para ajudar os usuários a validar snapshots existentes do v1beta1 e confirmar que são objetos de recurso válidos. O webhook de validação rotula automaticamente objetos snapshot inválidos e impede a criação de futuros objetos inválidos. O webhook de validação é implantado pelo Kubernetes orchestrator. Consulte as instruções para implantar o webhook de validação manualmente "aqui". Encontre exemplos de manifestos de instantâneos inválidos "aqui".

O exemplo detalhado abaixo explica as construções necessárias para trabalhar com snapshots e mostra como os snapshots podem ser criados e usados.

## Passo 1: Configure a. VolumeSnapshotClass

Antes de criar um instantâneo de volume, configure um xref:./trident-use/./Trident-reference/objects.html[VolumeSnapshotClass].

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

`driver`O ponto é o condutor CSI do Astra Trident. `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o instantâneo físico subjacente no cluster de armazenamento é retido mesmo quando o `VolumeSnapshot` objeto é excluído.

## Passo 2: Crie um instantâneo de um PVC existente

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

O instantâneo está sendo criado para um PVC chamado `pvcl`, e o nome do instantâneo é definido como `pvcl-snap`.

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

kubectl get volumesnapshots
NAME                AGE
pvcl-snap           50s
```

Isso criou um `VolumeSnapshot` objeto. Um `VolumeSnapshot` é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

É possível identificar o `VolumeSnapshotContent` objeto para o `pvcl-snap` `VolumeSnapshot` descrevendo-o.

```

kubect1 describe volumesnapshots pvcl-snap
Name:          pvcl-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:  pvcl-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvcl
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:  true
  Restore Size:  3Gi
.
.

```

O Snapshot Content Name identifica o objeto VolumeSnapshotContent que serve este instantâneo. O Ready To Use parâmetro indica que o instantâneo pode ser usado para criar um novo PVC.

### Etapa 3: Criar PVCs a partir do VolumeSnapshots

Veja o exemplo a seguir para criar um PVC usando um 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` Mostra que o PVC deve ser criado usando um `VolumeSnapshot` nomeado `pvc1-snap` como a fonte dos dados. Isso instrui o Astra Trident a criar um PVC a partir do snapshot. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.



Ao excluir um volume persistente com snapshots associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Para que o volume do Astra Trident seja excluído, os snapshots do volume devem ser removidos.

## Encontre mais informações

- ["Instantâneos de volume"](#)
- `xref:./trident-use/./Trident-reference/objects.html[VolumeSnapshotClass]`

## Expanda volumes

O Astra Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes depois que eles são criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI e NFS.

### Expanda um volume iSCSI

É possível expandir um iSCSI Persistent volume (PV) usando o provisionador de CSI.



A expansão de volume iSCSI é suportada pelos `ontap-san` `ontap-san-economy drivers` , , `solidfire-san` e requer o Kubernetes 1,16 e posterior.

#### Visão geral

A expansão de um iSCSI PV inclui os seguintes passos:

- Editando a definição `StorageClass` para definir o `allowVolumeExpansion` campo como `true`.
- Editar a definição de PVC e atualizar o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior que o tamanho original.
- A fixação do PV deve ser fixada a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:
  - Se o PV estiver conectado a um pod, o Astra Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
  - Ao tentar redimensionar um PV não anexado, o Astra Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

O exemplo abaixo mostra como funcionam os PVS iSCSI em expansão.

### Etapa 1: Configure o `StorageClass` para dar suporte à expansão de volume

```

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

```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

## Etapa 2: Crie um PVC com o StorageClass que você criou

```

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

```

O Astra Trident cria um volume persistente (PV) e o associa a essa reivindicação de volume 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

```



### Passo 3: Defina um pod que prende o PVC

Neste exemplo, é criado um pod que usa o `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
```

### Passo 4: Expanda o PV

Para redimensionar o PV que foi criado de 1Gi a 2Gi, edite a definição de PVC e atualize o `spec.resources.requests.storage` para 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
  ...
```

### **Etapas 5: Validar a expansão**

É possível validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e volume Astra Trident:

```

kubect1 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
kubect1 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    |
+-----+-----+-----+
+-----+-----+-----+-----+

```

## Expandir um volume NFS

O Astra Trident dá suporte à expansão de volume para PVS NFS provisionados em `ontap-nas` `ontap-nas-economy` , , , `ontap-nas-flexgroup` `gcp-cvs` e `azure-netapp-files` backends.

### Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão de volume definindo o `allowVolumeExpansion` campo para `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

```

Se você já criou uma classe de armazenamento sem essa opção, você pode simplesmente editar a classe de armazenamento existente usando `kubect1 edit storageclass` para permitir a expansão de volume.

## Etapa 2: Crie um PVC com o StorageClass que você criou

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

O Astra Trident deve criar um PV NFS de 20MiB para este PVC:

```
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
```

## Passo 3: Expanda o PV

Para redimensionar o 20MiB PV recém-criado para 1GiB, edite o PVC e defina `spec.resources.requests.storage` como 1GB:

```
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
  ...
```

#### **Etapa 4: Validar a expansão**

Você pode validar o redimensionamento trabalhado corretamente verificando o tamanho do PVC, PV e o volume Astra Trident:

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY    ACCESS MODES   STORAGECLASS   AGE
ontapnas20mb    Bound        pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7    1Gi
RWO                ontapnas                4m44s

kubect1 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 volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import`o .

### Drivers que suportam importação de volume

Esta tabela mostra os drivers que suportam a importação de volumes e a versão em que foram introduzidos.

Condutor	Solte
ontap-nas	19,04
ontap-nas-flexgroup	19,04
solidfire-san	19,04
azure-netapp-files	19,04

Conductor	Solte
gcp-cvs	19,04
ontap-san	19,04

## Por que devo importar volumes?

Existem vários casos de uso para importar um volume para o Trident:

- Containerizar um aplicativo e reutilizar seu conjunto de dados existente
- Usando um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruindo um cluster do Kubernetes com falha
- Migração de dados de aplicativos durante a recuperação de desastres

## Como funciona a importação?

O arquivo PVC (Persistent volume Claim) é usado pelo processo de importação de volume para criar o PVC. No mínimo, o arquivo PVC deve incluir os campos nome, namespace, accessModes e storageClassName como mostrado no exemplo a seguir.

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

O `tridentctl` cliente é usado para importar um volume de armazenamento existente. O Trident importa o volume persistindo metadados de volume e criando o PVC e o PV.

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

Para importar um volume de storage, especifique o nome do back-end do Astra Trident que contém o volume, bem como o nome que identifica exclusivamente o volume no storage (por exemplo: ONTAP FlexVol, Element volume, caminho de volume CVS). O volume de storage deve permitir acesso de leitura/gravação e ser acessível pelo back-end especificado do Astra Trident. O `-f` argumento string é necessário e especifica o caminho para o arquivo PVC YAML ou JSON.

Quando o Astra Trident recebe a solicitação de volume de importação, o tamanho do volume existente é determinado e definido no PVC. Depois que o volume é importado pelo driver de armazenamento, o PV é criado com uma ClaimRef para o PVC. A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com êxito o PVC e o PV, a política de recuperação é atualizada para

corresponder à política de recuperação da Classe de armazenamento. Se a política de recuperação da Classe de armazenamento for `delete`, o volume de armazenamento será excluído quando o PV for excluído.

Quando um volume é importado com o `--no-manage` argumento, o Trident não executa nenhuma operação adicional no PVC ou PV para o ciclo de vida dos objetos. Como o Trident ignora eventos PV e PVC para `--no-manage` objetos, o volume de armazenamento não é excluído quando o PV é excluído. Outras operações, como clone de volume e redimensionamento de volume, também são ignoradas. Essa opção é útil se você quiser usar o Kubernetes para workloads em contêineres, mas de outra forma quiser gerenciar o ciclo de vida do volume de storage fora do Kubernetes.

Uma anotação é adicionada ao PVC e ao PV que serve para um duplo propósito de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Esta anotação não deve ser modificada ou removida.

O Trident 19,07 e posterior lidam com a fixação de PVS e monta o volume como parte da importação. Para importações usando versões anteriores do Astra Trident, não haverá nenhuma operação no caminho de dados e a importação de volume não verificará se o volume pode ser montado. Se um erro for cometido com a importação de volume (por exemplo, o StorageClass está incorreto), você poderá recuperar alterando a política de recuperação no PV para `retain`, excluindo o PVC e o PV e tentando novamente o comando de importação de volume.

## ontap-nas e ontap-nas-flexgroup importações

Cada volume criado com o `ontap-nas` driver é um FlexVol no cluster do ONTAP. A importação do FlexVols com o `ontap-nas` driver funciona da mesma forma. Um FlexVol que já existe em um cluster ONTAP pode ser importado como `ontap-nas` PVC. Da mesma forma, os vols FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.



Um volume ONTAP deve ser do tipo `rw` a ser importado pelo Trident. Se um volume for do tipo `dp`, é um volume de destino SnapMirror; você deve quebrar a relação de espelhamento antes de importar o volume para o Trident.



O `ontap-nas` driver não pode importar e gerenciar `qtrees`. Os `ontap-nas` drivers e `ontap-nas-flexgroup` não permitem nomes de volume duplicados.

Por exemplo, para importar um volume nomeado `managed_volume` em um backend `ontap_nas` chamado , use o seguinte 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 um volume chamado `unmanaged_volume` (no `ontap_nas` backend), que o Trident não gerenciará, use o seguinte 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      |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

Ao usar o `--no-manage` argumento, o Trident não renomeará o volume nem validará se o volume foi montado. A operação de importação de volume falha se o volume não tiver sido montado manualmente.



Um bug existente anteriormente com a importação de volumes com `UnixPermissions` personalizados foi corrigido. Você pode especificar `unixPermissions` em sua definição de PVC ou configuração de back-end e instruir o Astra Trident a importar o volume de acordo.

## ontap-san importar

O Astra Trident também pode importar ONTAP SAN FlexVols que contenham um único LUN. Isso é consistente com o `ontap-san` driver, que cria um FlexVol para cada PVC e um LUN dentro do FlexVol. Você pode usar o `tridentctl import` comando da mesma forma que em outros casos:

- Inclua o nome `ontap-san` do backend.
- Forneça o nome do FlexVol que precisa ser importado. Lembre-se, este FlexVol contém apenas um LUN que deve ser importado.
- Fornecer o caminho da definição de PVC que deve ser usado com a `-f` bandeira.
- Escolha entre ter o PVC gerenciado ou não gerenciado. Por padrão, o Trident gerenciará o PVC e renomeará o FlexVol e o LUN no back-end. Para importar como um volume não gerenciado, passe o `--no-manage` sinalizador.



Ao importar um volume não gerenciado `ontap-san`, você deve certificar-se de que o LUN no FlexVol é nomeado `lun0` e é mapeado para um grupo com os iniciadores desejados. O Astra Trident trata isso automaticamente para uma importação gerenciada.

O Astra Trident irá então importar o FlexVol e associá-lo à definição de PVC. O Astra Trident também renomeia o FlexVol para `pvc-<uuid>` o formato e o LUN dentro do FlexVol para `lun0`.



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



O driver Element suporta nomes de volume duplicados. Se houver nomes de volume duplicados, o processo de importação de volume do Trident retornará um erro. Como solução alternativa, clone o volume e forneça um nome de volume exclusivo. Em seguida, importe o volume clonado.

## gcp-cvs importar



Para importar um volume com o suporte do NetApp Cloud Volumes Service no GCP, identifique o volume pelo caminho do volume em vez do nome.

Para importar um gcp-cvs volume no back-end chamado gcpcvs\_YEppr com o caminho de volume adroit-jolly-swift do , use o seguinte comando:

```
tridentctl import volume gcpcvs_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         |
+-----+-----+-----+
+-----+-----+-----+-----+
```



O caminho do volume é a parte do caminho de exportação do volume após :/. Por exemplo, se o caminho de exportação for 10.0.0.1:/adroit-jolly-swift, o caminho do volume será adroit-jolly-swift.

## azure-netapp-files **importar**

Para importar um azure-netapp-files volume no back-end chamado `azurenetaappfiles_40517` com o caminho do volume `importvoll1`, execute o seguinte comando:

```
tridentctl import volume azurenetaappfiles_40517 importvoll1 -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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



O caminho de volume para o volume do ANF está presente no caminho de montagem após `:/`. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvoll1`, o caminho do volume será `importvoll1`.

## Informações sobre direitos autorais

Copyright © 2025 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTES DOCUMENTOS. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALIENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTES SOFTWARES, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

## Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em <http://www.netapp.com/TM> são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.