



Provisionar e gerenciar volumes

Trident

NetApp
January 14, 2026

Índice

Provisionar e gerenciar volumes	1
Provisionar um volume	1
Visão geral	1
Crie o PV e o PVC	4
Expandir volumes	5
Expandir um volume iSCSI	5
Expandir um volume NFS	9
Importar volumes	12
Visão geral e considerações	12
Importar um volume	13
Exemplos	14
Personalizar nomes e rótulos de volume	19
Antes de começar	19
Limitações	19
Principais comportamentos de nomes de volume personalizáveis	19
Exemplos de configuração de backend com modelo de nome e rótulos	19
Exemplos de modelo de nome	21
Pontos a considerar	22
Compartilhar um volume NFS entre namespaces	22
Características	22
Início rápido	23
Configure os namespaces de origem e destino	24
Eliminar um volume compartilhado	25
`tridentctl get` Use para consultar volumes subordinados	25
Limitações	26
Para mais informações	26
Replique volumes usando o SnapMirror	26
Pré-requisitos de replicação	26
Crie um PVC espelhado	27
Estados de replicação de volume	30
Promover PVC secundário durante um failover não planejado	30
Promover PVC secundário durante um failover planejado	31
Restaurar uma relação de espelhamento após um failover	31
Operações adicionais	31
Atualizar relações de espelho quando o ONTAP estiver online	32
Atualizar relações de espelho quando o ONTAP estiver offline	32
Use a topologia CSI	33
Visão geral	33
Etapa 1: Crie um back-end com reconhecimento de topologia	34
Etapa 2: Defina StorageClasses que estejam cientes da topologia	36
Passo 3: Criar e usar um PVC	37
Atualize os backends para incluir <code>supportedTopologies</code>	40
Encontre mais informações	40

Trabalhar com instantâneos	40
Visão geral	40
Criar um instantâneo de volume	41
Crie um PVC a partir de um instantâneo de volume	42
Importar um instantâneo de volume	43
Recuperar dados de volume usando snapshots	45
Restauração de volume no local a partir de um instantâneo	45
Eliminar um PV com instantâneos associados	47
Implantar um controlador de snapshot de volume	47
Links relacionados	48

Provisionar e gerenciar volumes

Provisionar um volume

Crie um Persistentvolume (PV) e um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

Visão geral

A "[Persistentvolume](#)" (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O "[PersistentVolumeClaim](#)" (PVC) é um pedido de acesso ao Persistentvolume no cluster.

O PVC pode ser configurado para solicitar o armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que o Persistentvolume e o modo de acesso, como desempenho ou nível de serviço.

Depois de criar o PV e o PVC, você pode montar o volume em um pod.

Manifestos de amostra

Persistentvolume Sample MANIFEST

Este manifesto de exemplo mostra um PV básico de 10Gi que está associado ao StorageClass . basic-csi

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolumeClaim amostra manifestos

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWO associado a um StorageClass `basic-csi` chamado .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO associado a um StorageClass `protection-gold` chamado .

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

Amostras de manifesto POD

Estes exemplos mostram configurações básicas para anexar o PVC a um pod.

Configuração básica

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Configuração básica NVMe/TCP

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      resources: {}
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: pvc-san-nvme
```

Crie o PV e o PVC

Passos

1. Crie o PV.

```
kubectl create -f pv.yaml
```

2. Verifique o estado do PV.

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. Crie o PVC.

```
kubectl create -f pvc.yaml
```

4. Verifique o estado do PVC.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-storage	Bound	pv-name	2Gi	RWO		5m

5. Monte o volume num pod.

```
kubectl create -f pv-pod.yaml
```



Pode monitorizar o progresso utilizando `kubectl get pod --watch`o .

6. Verifique se o volume está montado no /my/mount/path.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

```
kubectl delete pod pv-pod
```

["Objetos Kubernetes e Trident"](#) Consulte para obter detalhes sobre como as classes de armazenamento interagem com os PersistentVolumeClaim parâmetros e para controlar como o Trident provisiona volumes.

Expanda volumes

O 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.

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

Edite a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.

```
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

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior do que o tamanho original.

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

O Trident cria um volume persistente (PV) e associa-o a esta reivindicação de volume persistente (PVC).

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY
san-pvc	Bound	pvc-8a814d62-bd58-4253-b0d1-82f2885db671	1Gi

```
kubectl get pv
```

NAME	RECLAIM POLICY	STATUS	CLAIM	CAPACITY	ACCESS MODES	AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671	Delete	Bound	default/san-pvc	1Gi	RWO	10s

Passo 3: Defina um pod que prende o PVC

Fixe o PV 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 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 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.

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

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
  ...
```

Etapa 5: Validar a expansão

Você pode validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e o volume 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 |
+-----+-----+-----+-----+
|          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 Trident oferece 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.

Etapas 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 `kubectl 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
```

Trident deve criar um PV NFS 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 1GiB:

```

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 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 volumes

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

Visão geral e considerações

Você pode importar um volume para o Trident para:

- Containerize um aplicativo e reutilize seu conjunto de dados existente
- Use um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster do Kubernetes com falha
- Migrar dados da aplicação durante a recuperação de desastre

Considerações

Antes de importar um volume, reveja as seguintes considerações.

- O Trident pode importar apenas volumes ONTAP do tipo RW (leitura-gravação). Os volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação de

espelhamento antes de importar o volume para o Trident.

- Sugerimos importar volumes sem conexões ativas. Para importar um volume usado ativamente, clonar o volume e, em seguida, executar a importação.



Isso é especialmente importante para volumes de bloco, já que o Kubernetes não sabia da conexão anterior e poderia facilmente anexar um volume ativo a um pod. Isso pode resultar em corrupção de dados.

- Embora `StorageClass` deva ser especificado em um PVC, o Trident não usa esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação de volume para selecionar os pools disponíveis com base nas características de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo se o volume existir em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- 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.
- Por padrão, o Trident gerencia o PVC e renomeia o `FlexVol` e o `LUN` no back-end. Você pode passar o `--no-manage` sinalizador para importar um volume não gerenciado. Se você usar `--no-manage`o` , o Trident não realiza nenhuma operação adicional no PVC ou no PV para o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído e 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.

Importar um volume

Pode utilizar `tridentctl import` para importar um volume.

Passos

1. Crie o arquivo PVC (Persistent volume Claim) (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O ficheiro PVC deve incluir `name namespace` , , `accessModes storageClassName` e . Opcionalmente, você pode especificar `unixPermissions` em sua definição de PVC.

O seguinte é um exemplo de uma especificação mínima:


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



Não inclua parâmetros adicionais, como nome PV ou tamanho do volume. Isso pode fazer com que o comando de importação falhe.

2. Use o `tridentctl import volume` comando para especificar o nome do back-end do Trident que contém o volume e o nome que identifica exclusivamente o volume no storage (por exemplo: ONTAP FlexVol, volume do elemento, caminho do Cloud Volumes Service). O `-f` argumento é necessário para especificar o caminho para o arquivo PVC.

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

Exemplos

Reveja os exemplos de importação de volume a seguir para drivers suportados.

ONTAP nas e ONTAP nas FlexGroup

O Trident suporta a importação de volume usando os `ontap-nas drivers` e `ontap-nas-flexgroup`.



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

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.

Exemplos de ONTAP nas

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

Volume gerenciado

O exemplo a seguir importa um volume nomeado `managed_volume` em um backend chamado `ontap_nas`:

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

NAME	SIZE	STORAGE CLASS
pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	1.0 GiB	standard
file	online	true

Volume não gerenciado

Ao usar o `--no-manage` argumento, o Trident não renomeará o volume.

O exemplo a seguir é importado `unmanaged_volume` `ontap_nas` no backend:

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

NAME	SIZE	STORAGE CLASS
pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	1.0 GiB	standard
file	online	false

San ONTAP

O Trident suporta a importação de volume usando os `ontap-san drivers` e `ontap-san-economy`.

O Trident 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. A Trident importa o FlexVol e associa-o à definição de PVC.

Exemplos de SAN ONTAP

A seguir mostra um exemplo de um volume gerenciado e uma importação de volume não gerenciado.

Volume gerenciado

Para volumes gerenciados, o Trident renomeia o FlexVol para `pvc-<uuid>` o formato e o LUN no FlexVol para `lun0`.

O exemplo a seguir importa `ontap-san-managed` o FlexVol que está presente no `ontap_san_default` back-end:

```
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
pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a	20 MiB	basic
block	cd394786-ddd5-4470-adc3-10c5ce4ca757	online

Volume não gerenciado

O exemplo a seguir é importado `unmanaged_example_volume` `ontap_san` no 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
pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	1.0 GiB	san-blog
block	e3275890-7d80-4af6-90cc-c7a0759f555a	online

Se você tiver LUNS mapeados para grupos que compartilham uma IQN com um nó Kubernetes IQN, como mostrado no exemplo a seguir, você receberá o erro: LUN already mapped to initiator(s) in this group. Você precisará remover o iniciador ou desmapear o LUN para importar o volume.

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

O Trident suporta o software NetApp Element e a importação de volume NetApp HCI usando o `solidfire-san` driver.



O driver Element suporta nomes de volume duplicados. No entanto, o Trident retorna um erro se houver nomes de volume duplicados. Como solução alternativa, clone o volume, forneça um nome de volume exclusivo e importe o volume clonado.

Exemplo de elemento

O exemplo a seguir importa um `element-managed` volume no backend `.element_default`

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

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          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

O Trident suporta a importação de volume usando `gcp-cvs` o driver.



Para importar um volume com o suporte do NetApp Cloud Volumes Service no Google Cloud Platform, identifique o volume pelo caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de exportação for `10.0.0.1:/adroit-jolly-swift`, o caminho do volume será `adroit-jolly-swift`.

Exemplo do Google Cloud Platform

O exemplo a seguir importa um gcp-cvs volume no back-end gcpcvs_YEppr com o caminho de volume adroit-jolly-swift do .

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

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

Azure NetApp Files

O Trident suporta a importação de volume usando azure-netapp-files o driver.



Para importar um volume Azure NetApp Files, identifique o volume pelo seu caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o :/. Por exemplo, se o caminho de montagem for 10.0.0.2:/importvol1, o caminho do volume será importvol1.

Exemplo de Azure NetApp Files

O exemplo a seguir importa um azure-netapp-files volume no back-end azurenetafiles_40517 com o caminho do volume importvol1 .

```
tridentctl import volume azurenetafiles_40517 importvol1 -f <path-to-
pvc-file> -n trident
```

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

Personalizar nomes e rótulos de volume

Com o Trident, você pode atribuir nomes e rótulos significativos aos volumes criados. Isso ajuda a identificar e mapear facilmente volumes para seus respectivos recursos do Kubernetes (PVCs). Você também pode definir modelos no nível de back-end para criar nomes de volume personalizados e rótulos personalizados; todos os volumes que você criar, importar ou clonar aderirão aos modelos.

Antes de começar

Nomes de volume e etiquetas personalizáveis suportam:

1. Operações de criação, importação e clonagem de volume.
2. No caso do driver ONTAP-nas-Economy, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
3. No caso do driver ONTAP-san-Economy, apenas o nome LUN está em conformidade com o modelo de nome.

Limitações

1. Os nomes de volume personalizáveis são compatíveis apenas com drivers locais da ONTAP.
2. Nomes de volume personalizáveis não se aplicam a volumes existentes.

Principais comportamentos de nomes de volume personalizáveis

1. Se ocorrer uma falha devido à sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se o aplicativo modelo falhar, o volume será nomeado de acordo com a convenção de nomenclatura existente.
2. O prefixo de armazenamento não é aplicável quando um volume é nomeado usando um modelo de nome da configuração de back-end. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com modelo de nome e rótulos

Modelos de nome personalizados podem ser definidos no nível raiz e/ou pool.

Exemplo de nível de raiz

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
      "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {"cluster": "ClusterA", "PVC":
    "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemplo de nível de pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels":{"labelname":"label1", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels":{"cluster":"label2", "name": "{{ .volume.Name }}"},
      "defaults":
      {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemplos de modelo de nome

Exemplo 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemplo 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}"
```


Pontos a considerar

1. No caso das importações de volume, as etiquetas são atualizadas apenas se o volume existente tiver etiquetas num formato específico. Por exemplo `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}:.`
2. No caso de importações de volume gerenciado, o nome do volume segue o modelo de nome definido no nível raiz na definição de back-end.
3. O Trident não suporta a utilização de um operador de corte com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, o Trident adicionará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume de economia nas exceder 64 caracteres de comprimento, o Trident nomeará os volumes de acordo com a convenção de nomenclatura existente. Para todos os outros drivers ONTAP, se o nome do volume exceder o limite de nome, o processo de criação de volume falhará.

Compartilhar um volume NFS entre namespaces

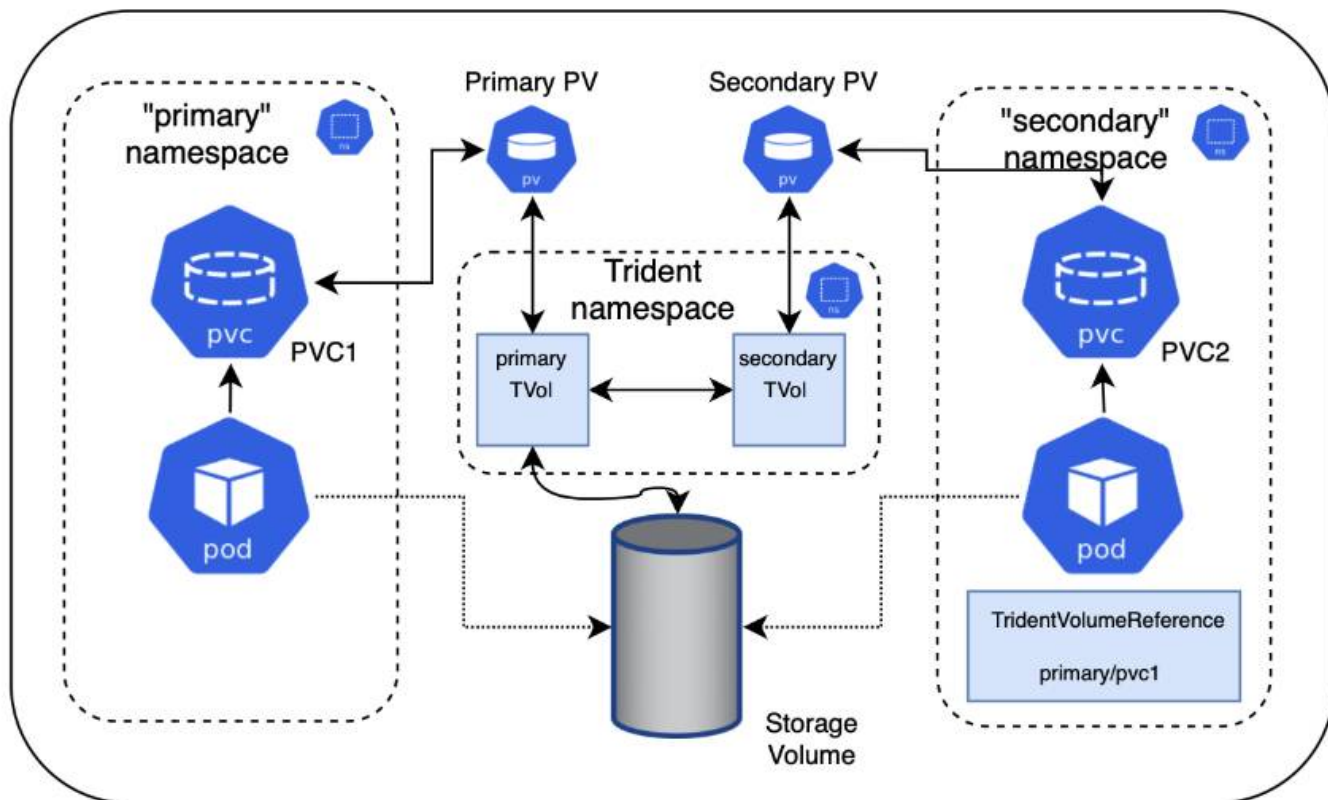
Usando o Trident, você pode criar um volume em um namespace principal e compartilhá-lo em um ou mais namespaces secundários.

Caraterísticas

O `TridentVolumeReference` CR permite que você compartilhe com segurança volumes NFS `ReadWriteMany` (RWX) em um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes tem os seguintes benefícios:

- Vários níveis de controle de acesso para garantir a segurança
- Funciona com todos os drivers de volume Trident NFS
- Não há dependência do `tridentctl` ou de qualquer outro recurso do Kubernetes não nativo

Este diagrama ilustra o compartilhamento de volumes NFS em dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volume NFS em apenas algumas etapas.

1

Configure o PVC de origem para compartilhar o volume

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2

Conceda permissão para criar um CR no namespace de destino

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o `CredentVolumeReference` CR.

3

Crie `TridentVolumeReference` no namespace de destino

O proprietário do namespace de destino cria o `TridentVolumeReference` CR para se referir ao PVC de origem.

4

Crie o PVC subordinado no namespace de destino

O proprietário do namespace de destino cria o PVC subordinado para usar a fonte de dados do PVC de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, o compartilhamento entre namespace requer colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função de usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a `shareToNamespace` anotação.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

A Trident cria o PV e seu volume de storage NFS no back-end.



- Você pode compartilhar o PVC para vários namespaces usando uma lista delimitada por vírgulas. Por exemplo, `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/shareToNamespace: *`
- Você pode atualizar o PVC para incluir a `shareToNamespace` anotação a qualquer momento.

2. **Cluster admin:** Crie a função personalizada e kubeconfig para conceder permissão ao proprietário do namespace de destino para criar o `TridentVolumeReference` CR no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um `CredentVolumeReference` CR no namespace de destino que se refere ao namespace de origem `pvc1`.

```

apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1

```

4. **Proprietário do namespace de destino:** Crie um PVC (pvc2) no namespace de destino (namespace2) usando a shareFromPVC anotação para designar o PVC de origem.

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi

```



O tamanho do PVC de destino deve ser inferior ou igual ao PVC de origem.

Resultados

O Trident lê a shareFromPVC anotação no PVC de destino e cria o PV de destino como um volume subordinado sem recurso de armazenamento próprio que aponta para o PV de origem e compartilha o recurso de armazenamento PV de origem. O PVC e o PV de destino aparecem encadernados normalmente.

Eliminar um volume partilhado

Você pode excluir um volume compartilhado entre vários namespaces. O Trident removerá o acesso ao volume no namespace de origem e manterá o acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, o Trident exclui o volume.

`tridentctl get` Use para consultar volumes subordinados

Usando o `tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para obter mais informações, consulte o `tridentctl` [comandos e opções](#).

```
Usage:
  tridentctl get [option]
```

Bandeiras -

- ``-h, --help`: Ajuda para volumes.
- `--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
- `--subordinateOf string`: Limitar consulta a subordinados de volume.

Limitações

- O Trident não pode impedir que namespaces de destino gravem no volume compartilhado. Você deve usar o bloqueio de arquivos ou outros processos para evitar a substituição de dados de volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` anotações ou `shareFromNamespace` excluindo o `TridentVolumeReference` CR. Para revogar o acesso, você deve excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para mais informações

Para saber mais sobre o acesso ao volume entre namespace:

- ["Compartilhamento de volumes entre namespaces: Diga olá ao acesso ao volume entre namespace"](#) Visite [.NetAppTV](#).
- Assista à demonstração no ["NetAppTV"](#).

Replique volumes usando o SnapMirror

O Trident oferece suporte a relacionamentos espelhados entre um volume de origem em um cluster e o volume de destino no cluster com peering para replicação de dados para recuperação de desastres. Você pode usar uma Definição de recursos personalizados (CRD) para executar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova as relações de espelho entre volumes
- Quebre as relações do espelho
- Promover o volume secundário durante as condições de desastre (failovers)
- Realizar a transição sem perda de aplicativos do cluster para o cluster (durante failovers planejados ou migrações)

Pré-requisitos de replicação

Certifique-se de que os seguintes pré-requisitos sejam atendidos antes de começar:

Clusters de ONTAP

- **Trident:** O Trident versão 22,10 ou posterior deve existir nos clusters do Kubernetes de origem e destino que utilizam o ONTAP como um back-end.
- **Licenças:** As licenças assíncronas do ONTAP SnapMirror usando o pacote proteção de dados devem estar ativadas nos clusters ONTAP de origem e destino. ["Visão geral do licenciamento do SnapMirror no ONTAP"](#) Consulte para obter mais informações.

Peering

- **Cluster e SVM:** Os backends de storage do ONTAP devem ser colocados em Contato. ["Visão geral do peering de cluster e SVM"](#) Consulte para obter mais informações.



Certifique-se de que os nomes do SVM usados na relação de replicação entre dois clusters ONTAP sejam exclusivos.

- **Trident e SVM:** Os SVMs remotas com peering devem estar disponíveis para o Trident no cluster de destino.

Drivers suportados

- A replicação de volume é compatível com os drivers ONTAP-nas e ONTAP-san.

Crie um PVC espelhado

Siga estas etapas e use os exemplos CRD para criar relação de espelhamento entre volumes primário e secundário.

Passos

1. Execute as etapas a seguir no cluster primário do Kubernetes:
 - a. Crie um objeto StorageClass com o `trident.netapp.io/replication: true` parâmetro.

Exemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

- b. Crie um PVC com StorageClass criado anteriormente.

Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

- c. Crie um MirrorRelationship CR com informações locais.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
    - localPVCName: csi-nas
```

O Trident obtém as informações internas para o volume e o estado atual de proteção de dados (DP) do volume e, em seguida, preenche o campo de status do espelhamento.

- d. Obtenha o tridentMirrorRelationship CR para obter o nome interno e SVM do PVC.

```
kubect1 get tmr csi-nas
```

```

kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
    - localPVCName: csi-nas
status:
  conditions:
    - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1

```

2. Execute as etapas a seguir no cluster secundário do Kubernetes:

- a. Crie um StorageClass com o parâmetro Trident.NetApp.io/replicação: True.

Exemplo

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true

```

- b. Crie um MirrorRelationship CR com informações de destino e origem.

Exemplo

```

kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
    - localPVCName: csi-nas
      remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"

```


O Trident criará um relacionamento SnapMirror com o nome da política de relacionamento configurado (ou padrão para o ONTAP) e inicializá-lo.

- c. Crie um PVC com StorageClass criado anteriormente para atuar como secundário (destino SnapMirror).

Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
    - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

O Trident verificará se há CRD de relacionamento de espelhamento e não criará o volume se a relação não existir. Se o relacionamento existir, o Trident garantirá que o novo FlexVol volume seja colocado em um SVM que seja emparelhado com o SVM remoto definido no espelhamento.

Estados de replicação de volume

Um relacionamento de espelhamento do Trident (TMR) é um CRD que representa um fim de uma relação de replicação entre PVCs. O TMR de destino tem um estado, que informa ao Trident qual é o estado desejado. O TMR de destino tem os seguintes estados:

- *** Estabelecido***: O PVC local é o volume de destino de uma relação de espelho, e esta é uma nova relação.
- **Promovido**: O PVC local é ReadWrite e montável, sem relação de espelho atualmente em vigor.
- *** Restabelecido***: O PVC local é o volume de destino de uma relação de espelho e também estava anteriormente nessa relação de espelho.
 - O estado restabelecido deve ser usado se o volume de destino estiver em uma relação com o volume de origem, porque ele sobrescreve o conteúdo do volume de destino.
 - O estado restabelecido falhará se o volume não estiver previamente em uma relação com a fonte.

Promover PVC secundário durante um failover não planejado

Execute a seguinte etapa no cluster secundário do Kubernetes:

- Atualize o campo `spec.State` do `TridentMirrorRelationship` para `promoted`.

Promover PVC secundário durante um failover planejado

Durante um failover planejado (migração), execute as seguintes etapas para promover o PVC secundário:

Passos

1. No cluster primário do Kubernetes, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster principal do Kubernetes, crie o SnapshotInfo CR para obter detalhes internos.

Exemplo

```
kind: SnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshot-name: csi-nas-snapshot
```

3. No cluster secundário do Kubernetes, atualize o campo *spec.State* do *tridentMirrorRelationship* CR para *promoted* e *spec.promotedSnapshotHandle* para ser o *internalName* do snapshot.
4. No cluster secundário do Kubernetes, confirme o status (campo *status.State*) do *TridentMirrorRelationship* para promovido.

Restaurar uma relação de espelhamento após um failover

Antes de restaurar uma relação de espelho, escolha o lado que você deseja fazer como o novo primário.

Passos

1. No cluster secundário do Kubernetes, certifique-se de que os valores do campo *spec.remoteVolumeHandle* no *TridentMirrorRelationship* sejam atualizados.
2. No cluster secundário do Kubernetes, atualize o campo *spec.mirror* do *TridentMirrorRelationship* para *reestablished*.

Operações adicionais

O Trident dá suporte às seguintes operações nos volumes primário e secundário:

Replique PVC primário para um novo PVC secundário

Certifique-se de que você já tem um PVC primário e um PVC secundário.

Passos

1. Exclua as CRDs *PersistentVolumeClaim* e *TridentMirrorRelationship* do cluster secundário (destino) estabelecido.
2. Exclua o CRD do *tridentMirrorRelationship* do cluster primário (de origem).
3. Crie um novo CRD de *TridentMirrorRelationship* no cluster primário (de origem) para o novo PVC secundário (de destino) que você deseja estabelecer.

Redimensione um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado como normal, o ONTAP irá expandir automaticamente qualquer destino flexvol se a quantidade de dados exceder o tamanho atual.

Remova a replicação de um PVC

Para remover a replicação, execute uma das seguintes operações no volume secundário atual:

- Exclua o MirrorRelationship no PVC secundário. Isso quebra a relação de replicação.
- Ou atualize o campo spec.State para *promovido*.

Excluir um PVC (que foi anteriormente espelhado)

O Trident verifica se há PVCs replicados e libera a relação de replicação antes de tentar excluir o volume.

Eliminar um TMR

A exclusão de um TMR em um lado de um relacionamento espelhado faz com que o TMR restante passe para o estado *promovido* antes que o Trident conclua a exclusão. Se o TMR selecionado para exclusão já estiver no estado *promovido*, não há relacionamento de espelhamento existente e o TMR será removido e o Trident promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados do SnapMirror para o volume local no ONTAP. Se este volume for usado em uma relação de espelho no futuro, ele deve usar um novo TMR com um estado de replicação de volume *established* ao criar a nova relação de espelho.

Atualizar relações de espelho quando o ONTAP estiver online

As relações de espelho podem ser atualizadas a qualquer momento depois que são estabelecidas. Pode utilizar os `state: promoted` campos ou `state: reestablished` para atualizar as relações. Ao promover um volume de destino para um volume ReadWrite regular, você pode usar *promotedSnapshotHandle* para especificar um snapshot específico para restaurar o volume atual.

Atualizar relações de espelho quando o ONTAP estiver offline

Você pode usar um CRD para executar uma atualização do SnapMirror sem que o Trident tenha conectividade direta com o cluster do ONTAP. Consulte o seguinte formato de exemplo do TridentActionMirrorUpdate:

Exemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` Reflete o estado do CRD do TridentActionMirrorUpdate. Ele pode tomar um valor de *successful*, *in progress* ou *Failed*.

Use a topologia CSI

O Trident pode criar e anexar volumes seletivamente a nós presentes em um cluster do Kubernetes, utilizando o ["Recurso de topologia CSI"](#).

Visão geral

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 Trident usa a topologia de CSI.



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

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- Com `VolumeBindingMode` definido como `Immediate`, o 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 de Kubernetes executando um ["Versão do Kubernetes compatível"](#)

```
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"}
```

- 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 Trident seja instalado para que o 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","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"}]
```

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

Os back-ends de storage do Trident podem ser projetados para provisionar volumes seletivamente de acordo com as zonas de disponibilidade. Cada back-end pode transportar um bloco opcional `supportedTopologies` que representa uma lista de zonas e regiões com suporte. 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.

Aqui está um exemplo de definição 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` 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 Trident cria 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: 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
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  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 `workload` são rótulos 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 cria 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: 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 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-existent 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

Os snapshots de volume do Kubernetes de volumes persistentes (PVS) permitem cópias pontuais de volumes. Você pode criar um snapshot de um volume criado usando o Trident, importar um snapshot criado fora do Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volume de snapshots.

Visão geral

O instantâneo de volume é suportado por `ontap-nas`, `ontap-nas-flexgroup` , , `ontap-san`, `ontap-san-economy` , , , `solidfire-san` `gcp-cvs`, e `azure-netapp-files` drivers.

Antes de começar

Você deve ter um controlador de snapshot externo e definições personalizadas de recursos (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, [Implantar um controlador de snapshot de volume](#) consulte .



Não crie um controlador de snapshot se estiver criando instantâneos de volume sob demanda em um ambiente GKE. O GKE usa um controlador instantâneo oculto integrado.

Criar um instantâneo de volume

Passos

1. Criar um `VolumeSnapshotClass`. para obter mais informações, "[VolumeSnapshotClass](#)" consulte .
 - Os driver pontos para o driver Trident CSI.
 - `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.

Exemplo

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

2. Crie um instantâneo de um PVC existente.

Exemplos

- Este exemplo cria um instantâneo de um PVC existente.

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

- Este exemplo cria um objeto instantâneo de volume para um PVC chamado `pvc1` e o nome do instantâneo é definido como `pvc1-snap`. Um `VolumeSnapshot` é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

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

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

- Pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` `VolumeSnapshot` descrevendo-o. 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.

```
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
.
.
```

Crie um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um `VolumeSnapshot` nomeado `<pvc-name>` como a fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.



O PVC será criado no mesmo backend que o volume de origem. ["KB: A criação de um PVC a partir de um instantâneo de PVC do Trident não pode ser criada em um back-end alternativo"](#) Consulte a .

O exemplo a seguir cria o PVC usando `pvc1-snap` como fonte de dados.

```
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
```

Importar um instantâneo de volume

O Trident oferece suporte ao ["Processo de snapshot pré-provisionado do Kubernetes"](#) para permitir que o administrador de cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Trident.

Antes de começar

O Trident deve ter criado ou importado o volume pai do instantâneo.

Passos

1. **Cluster admin:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot de back-end. Isso inicia o fluxo de trabalho de snapshot no Trident.
 - Especifique o nome do instantâneo de back-end em annotations as `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
 - Especifique `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle`. esta é a única informação fornecida ao Trident pelo snapshotter externo na `ListSnapshots` chamada.



O `<volumeSnapshotContentName>` nem sempre pode corresponder ao nome do instantâneo do back-end devido a restrições de nomenclatura CR.

Exemplo

O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência a snapshot de back-end `snap-01`.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default

```

2. **Cluster admin:** Crie o VolumeSnapshot CR que faz referência ao VolumeSnapshotContent objeto. Isso solicita acesso para usar o VolumeSnapshot em um namespace dado.

Exemplo

O exemplo a seguir cria um VolumeSnapshot CR chamado import-snap que faz referência ao VolumeSnapshotContent import-snap-content chamado .

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. * Processamento interno (nenhuma ação necessária):* o Snapshotter externo reconhece o recém-criado VolumeSnapshotContent e executa a ListSnapshots chamada. Trident cria o TridentSnapshot.
 - O snapshotter externo define VolumeSnapshotContent para readyToUse e VolumeSnapshot para true.
 - Trident retorna readyToUse=true.
4. **Qualquer usuário:** Crie um PersistentVolumeClaim para fazer referência ao novo VolumeSnapshot, onde o spec.dataSource nome (ou spec.dataSourceRef) é o VolumeSnapshot nome.

Exemplo

O exemplo a seguir cria um PVC referenciando o VolumeSnapshot nome `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recuperar dados de volume usando snapshots

O diretório instantâneo é oculto por padrão para facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas` drivers e `ontap-nas-economy`. Ative o `.snapshot` diretório para recuperar dados de instantâneos diretamente.

Use a CLI do ONTAP de restauração de snapshot de volume para restaurar um volume para um estado gravado em um snapshot anterior.

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



Quando você restaura uma cópia snapshot, a configuração de volume existente é sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

Restauração de volume no local a partir de um instantâneo

O Trident fornece restauração rápida de volume no local a partir de um instantâneo usando o `TridentActionSnapshotRestore` CR (TASR). Esse CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

O Trident oferece suporte à restauração de snapshot no `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`, `google-cloud-netapp-volumes`, e `solidfire-san` drivers.

Antes de começar

Você deve ter um PVC vinculado e instantâneo de volume disponível.

- Verifique se o status do PVC está vinculado.

```
kubectl get pvc
```

- Verifique se o instantâneo do volume está pronto para ser usado.

```
kubectl get vs
```

Passos

1. Crie o TASR CR. Este exemplo cria um CR para instantâneo de PVC `pvc1` e volume `pvc1-snapshot`.



O TAR CR deve estar num espaço de nomes onde o PVC e VS existam.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

1. Aplique o CR para restaurar a partir do instantâneo. Este exemplo restaura do instantâneo `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

O Trident restaura os dados do snapshot. Você pode verificar o status de restauração de snapshot.

```
kubectl get tasr -o yaml

apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```



- Na maioria dos casos, o Trident não tentará automaticamente a operação em caso de falha. Terá de efetuar novamente a operação.
- Os usuários do Kubernetes sem acesso de administrador podem ter permissão para que o administrador crie um TASR CR em seu namespace de aplicativo.

Eliminar um PV com instantâneos associados

Ao excluir um volume persistente com snapshots associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Remova os instantâneos de volume para excluir o volume Trident.

Implantar um controlador de snapshot de volume

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

Passos

1. Criar CRDs de instantâneos de volume.

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

2. Crie o controlador instantâneo.

```
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
```



Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- ["Instantâneos de volume"](#)
- ["VolumeSnapshotClass"](#)

Informações sobre direitos autorais

Copyright © 2026 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 SOBRESSAIENTES; 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.