Provisionar e gerenciar volumes

Trident

NetApp
January 15, 2026

Índice

Provisionar e gerenciar volumes . . . 1
- Forneça um volume . . . 1
  - Visão geral . . . 1
  - Criar o PVC . . . 1
- Expandir volumes . . . 4
  - Expandir um volume iSCSI . . . 4
  - Expandir um volume FC . . . 8
  - Expandir um volume NFS . . . 12
- Volumes de importação . . . 15
  - Visão geral e considerações . . . 15
  - Importar um volume . . . 16
  - Exemplos . . . 17
- Personalize os nomes e rótulos dos volumes. . . . 23
  - Antes de começar . . . 23
  - Limitações . . . 23
  - Principais comportamentos dos nomes de volume personalizáveis . . . 23
  - Exemplos de configuração de backend com modelo de nome e rótulos . . . 24
  - Exemplos de modelos de nome . . . 25
  - Pontos a considerar . . . 26
- Compartilhar um volume NFS entre namespaces . . . 26
  - Características . . . 26
  - Início rápido . . . 27
  - Configure os namespaces de origem e destino. . . . 28
  - Excluir um volume compartilhado . . . 29
  - Usar `tridentctl get` para consultar volumes subordinados . . . 29
  - Limitações . . . 30
  - Para maiores informações . . . 30
- Clonar volumes entre namespaces . . . 30
  - Pré-requisitos . . . 30
  - Início rápido . . . 30
  - Configure os namespaces de origem e destino. . . . 31
  - Limitações . . . 33
- Replicar volumes usando o SnapMirror . . . 33
  - Pré-requisitos para replicação . . . 33
  - Crie um PVC espelhado . . . 34
  - Estados de replicação de volume . . . 37
  - Promover o PVC secundário durante uma falha não planejada. . . . 37
  - Promover o PVC secundário durante uma falha planejada . . . 37
  - Restaurar uma relação de espelhamento após uma falha. . . . 38
  - Operações adicionais . . . 38
  - Atualize os relacionamentos de espelhamento quando o ONTAP estiver online. . . . 39
  - Atualizar relações de espelhamento quando o ONTAP estiver offline . . . 39
- Utilizar a topologia CSI . . . 39

Visão geral . . . 39
Passo 1: Crie um backend com reconhecimento de topologia . . . 41
Etapa 2: Defina StorageClasses que levem em consideração a topologia. . . . 43
Etapa 3: Criar e usar um PVC . . . 44
Atualizar os backends para incluir `supportedTopologies` . . . 47
Encontre mais informações . . . 47
Trabalhar com instantâneos . . . 47
Visão geral . . . 47
Criar um instantâneo de volume . . . 48
Criar um PVC a partir de um instantâneo de volume . . . 49
Importar um instantâneo de volume . . . 50
Recupere dados de volume usando snapshots . . . 52
Restauração de volume in-place a partir de um snapshot . . . 52
Excluir um PV com snapshots associados . . . 54
Implante um controlador de instantâneo de volume . . . 54
Links relacionados . . . 55
Trabalhar com snapshots de grupos de volumes . . . 55
Criar snapshots de grupos de volumes . . . 56
Recupere dados de volume usando um instantâneo de grupo. . . . 57
Restauração de volume in-place a partir de um snapshot . . . 58
Excluir um PV com snapshots de grupo associados . . . 58
Implante um controlador de instantâneo de volume . . . 58
Links relacionados . . . 59

Provisionar e gerenciar volumes

Forneça um volume

Crie um PersistentVolumeClaim (PVC) que utilize a StorageClass do Kubernetes configurada para solicitar acesso ao PV. Em seguida, você pode montar o painel fotovoltaico em um suporte.

Visão geral

UM "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster.

O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Utilizando a StorageClass associada, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume, como o desempenho ou o nível de serviço.

Após criar o tubo de PVC, você pode montar o volume em um compartimento.

Criar o PVC

Passos

1. Crie o PVC.

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

2. Verifique o status do PVC.

```
kubectl get pvc
```

```
NAME         STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage  Bound    pv-name    1Gi        RWO                    5m
```

1. Instale o volume em um pod.

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

> Você pode monitorar o progresso usando `kubectl get pod --watch`.

2. Verifique se o volume está montado em `/my/mount/path`.

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

3. Agora você pode excluir o Pod. O aplicativo Pod deixará de existir, mas o volume permanecerá.

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

Exemplos de manifestos

Manifestações de exemplo de PersistentVolumeClaim

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

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

```
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 uma StorageClass chamada `protection-gold`.

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

Exemplos de manifesto de pod

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

Configuração básica

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

Configuração básica de NVMe/TCP

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Consulte"Objetos Kubernetes e Trident" Para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e parâmetros para controlar como o Trident provisiona volumes.

Expandir 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, NFS, SMB, NVMe/TCP e FC.

Expandir um volume iSCSI

Você pode expandir um Volume Persistente (PV) iSCSI usando o provisionador CSI.

> A expansão de volume iSCSI é suportada pelo `ontap-san` , `ontap-san-economy` , `solidfire-san` requer drivers e Kubernetes 1.16 ou posterior.

Etapa 1: Configure a classe de armazenamento para suportar a expansão de volume.

Edite a definição da classe de armazenamento para definir o `allowVolumeExpansion` campo para `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 uma StorageClass já existente, edite-a para incluir o `allowVolumeExpansion` parâmetro.

Passo 2: Crie um PVC com a StorageClass que você criou.

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

A Trident cria um Volume Persistente (PV) e o associa a esta 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
```

Etapa 3: Defina um compartimento que conecte o PVC.

Conecte o PV a um pod para que ele possa ser redimensionado. Existem dois cenários ao redimensionar um PV iSCSI:

- Se o PV estiver conectado a um pod, o Trident expande o volume no backend de armazenamento, verifica novamente o dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, o Trident expande o volume no servidor de armazenamento. Após o PVC ser vinculado a um pod, o Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da operação de expansão.

Neste exemplo, é criado um pod que utiliza 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
```

Etapa 4: Expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do 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: Valide a expansão

Você pode verificar se a expansão funcionou corretamente conferindo o tamanho do PVC, do PV e o volume do Trident :

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

Expandir um volume FC

Você pode expandir um Volume Persistente (PV) FC usando o provisionador CSI.

> ⓘ A expansão do volume FC é suportada pelo `ontap-san` O driver requer Kubernetes 1.16 ou posterior.

Etapa 1: Configure a classe de armazenamento para suportar a expansão de volume.

Edite a definição da classe de armazenamento para definir o `allowVolumeExpansion` campo para `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 uma StorageClass já existente, edite-a para incluir o `allowVolumeExpansion` parâmetro.

Passo 2: Crie um PVC com a StorageClass que você criou.

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

A Trident cria um Volume Persistente (PV) e o associa a esta 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
```

Etapa 3: Defina um compartimento que conecte o PVC.

Conecte o PV a um pod para que ele possa ser redimensionado. Existem dois cenários ao redimensionar um painel fotovoltaico de célula de combustível:

- Se o PV estiver conectado a um pod, o Trident expande o volume no backend de armazenamento, verifica novamente o dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, o Trident expande o volume no servidor de armazenamento. Após o PVC ser vinculado a um pod, o Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da

operação de expansão.

Neste exemplo, é criado um pod que utiliza 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
```

Etapa 4: Expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do 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: Valide a expansão

Você pode verificar se a expansão funcionou corretamente conferindo o tamanho do PVC, do PV e o volume do Trident :

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

Expandir um volume NFS

O Trident suporta a expansão de volume para PVs NFS provisionados em `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `gcp-cvs` , e `azure-netapp-files` back-ends.

Etapa 1: Configure a classe de armazenamento para suportar a expansão de volume.

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão do volume, definindo o seguinte: `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, basta editar a classe de armazenamento

existente usando `kubectl edit storageclass` para permitir a expansão do volume.

Passo 2: Crie um PVC com a StorageClass que você criou.

```
cat pvc-ontapnas.yaml
```

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

A Trident deve criar um PV NFS de 20 MiB 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
```

Etapa 3: Expandir o PV

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

```
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: Valide a expansão

Você pode verificar se o redimensionamento funcionou corretamente conferindo o tamanho do PVC, do PV e do 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volumes de importação

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

Visão geral e considerações

Você pode importar um volume para o Trident para:

- Containerize uma aplicação e reutilize seu conjunto de dados existente.
- Utilize um clone de um conjunto de dados para uma aplicação efêmera.
- Reconstruir um cluster Kubernetes com falha
- Migrar dados de aplicativos durante a recuperação de desastres

Considerações

Antes de importar um volume, revise as seguintes considerações.

- O Trident só pode importar volumes ONTAP do tipo RW (leitura e 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 que esteja sendo usado ativamente, clone o volume e, em seguida, execute a importação.

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

- No entanto `StorageClass` Deve ser especificado em um PVC; o Trident não usa esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação de volumes para selecionar entre os pools disponíveis com base nas características de armazenamento. Como o volume já existe, não é necessário selecionar um pool durante a importação. Portanto, a importação não falhará mesmo que o volume exista em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- O volume existente é determinado e definido no PVC. Após o volume ser importado pelo driver de armazenamento, o PV é criado com uma referência ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Após o Kubernetes vincular com sucesso 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 volume e o LUN no backend. Você pode passar o `--no-manage` sinalizador para importar um volume não gerenciado. Se você usar `--no-manage` A Trident não realiza nenhuma operação adicional no PVC ou PV durante o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído, e outras operações como clonagem e redimensionamento de volume também são ignoradas.

> Essa opção é útil se você deseja usar o Kubernetes para cargas de trabalho em contêineres, mas, fora isso, prefere gerenciar o ciclo de vida do volume de armazenamento fora do Kubernetes.

- Uma anotação é adicionada ao PVC e ao PV com a dupla função de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Esta anotação não deve ser modificada nem removida.

Importar um volume

Você pode usar `tridentctl import` Importar um volume.

Passos

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

   Segue abaixo um exemplo de 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 do PV ou tamanho do volume. Isso pode fazer com que o comando de importação falhe.

2. Use o `tridentctl import` Comando para especificar o nome do backend Trident que contém o volume e o nome que identifica exclusivamente o volume no armazenamento (por exemplo: ONTAP FlexVol, Element Volume, caminho do Cloud Volumes Service). O `-f` É necessário fornecer o argumento para especificar o caminho para o arquivo PVC.

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

Exemplos

Consulte os exemplos de importação de volume a seguir para obter informações sobre os drivers compatíveis.

ONTAP NAS e ONTAP NAS FlexGroup

O Trident suporta a importação de volumes usando o `ontap-nas` e `ontap-nas-flexgroup` motoristas.

- O Trident não suporta importação de volume usando o `ontap-nas-economy` motorista.
- O `ontap-nas` e `ontap-nas-flexgroup` Os drivers não permitem nomes de volume duplicados.

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

Exemplos de ONTAP NAS

A seguir, apresentamos um exemplo de importação de um volume gerenciado e de um volume não gerenciado.

Volume gerenciado

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

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

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

Volume não gerenciado

Ao usar o `--no-manage` argumento: Trident não renomeia o volume.

O exemplo a seguir importa `unmanaged_volume` no `ontap_nas` backend:

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

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

ONTAP SAN

O Trident suporta importação de volume usando o `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` motoristas.

O Trident pode importar volumes ONTAP SAN FlexVol que contêm um único LUN. Isto é consistente com o `ontap-san` driver, que cria um FlexVol volume para cada PVC e um LUN dentro do FlexVol volume. O Trident importa o FlexVol volume e o associa à definição de PVC. Trident pode importar `ontap-san-economy`

volumes que contêm vários LUNs.

Exemplos de SAN ONTAP

A seguir, apresentamos um exemplo de importação de um volume gerenciado e de um volume não gerenciado.

Volume gerenciado

Para volumes gerenciados, o Trident renomeia o FlexVol volume para `pvc-<uuid>` formato e o LUN dentro do FlexVol volume para `lun0` .

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

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

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

Volume não gerenciado

O exemplo a seguir importa `unmanaged_example_volume` no `ontap_san` backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se você tiver LUNs mapeadas para igroups que compartilham um IQN com o IQN de um nó do Kubernetes,

como mostrado no exemplo a seguir, você receberá o seguinte erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desvincular 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 oferece suporte ao software NetApp Element e à importação de volumes NetApp HCI usando o `solidfire-san` motorista.

> 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 exclusivo para o volume 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 |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Plataforma Google Cloud

O Trident suporta a importação de volumes usando o `gcp-cvs` motorista.

Para importar um volume com suporte do NetApp Cloud Volumes Service no Google Cloud Platform, identifique o volume pelo seu caminho. O caminho do volume é a porção 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 é `adroit-jolly-swift` .

Exemplo do Google Cloud Platform

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

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

Azure NetApp Files

O Trident suporta a importação de volumes usando o `azure-netapp-files` motorista.

Para importar um volume do Azure NetApp Files , identifique o volume pelo seu caminho. O caminho do volume é a porção 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 é `importvol1` .

Exemplo de Azure NetApp Files

O exemplo a seguir importa um `azure-netapp-files` volume no backend `azurenetappfiles_40517` com o caminho do volume `importvol1` .

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

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

Google Cloud NetApp Volumes

O Trident suporta a importação de volumes usando o `google-cloud-netapp-volumes` motorista.

Exemplo de Google Cloud NetApp Volumes

O exemplo a seguir importa um `google-cloud-netapp-volumes` volume no backend `backend-tbc-gcnv1` com o volume `testvoleasiaeast1` .

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

O exemplo a seguir importa um `google-cloud-netapp-volumes` Volume quando dois volumes estão presentes na mesma região:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                  NAME                    |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personalize os nomes e rótulos dos volumes.

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

Antes de começar

Suporte para nomes e rótulos de volume personalizáveis:

1. Operações de criação, importação e clonagem de volumes.
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 ONTAP locais.
2. Os nomes de volume personalizáveis não se aplicam a volumes existentes.

Principais comportamentos dos nomes de volume personalizáveis

1. Se ocorrer uma falha devido a uma sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se a aplicação do modelo falhar, o volume será nomeado de acordo com a convenção

de nomenclatura existente.

2. O prefixo de armazenamento não se aplica quando um volume é nomeado usando um modelo de nome da configuração do backend. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com modelo de nome e rótulos

É possível definir modelos de nomes personalizados no nível raiz e/ou no nível do pool.

Exemplo de nível 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.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemplo de nível de piscina

```
{
  "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 modelos 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 de importações em volume, os rótulos são atualizados somente se o volume existente possuir rótulos em um formato específico. Por exemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. No caso de importações de volumes gerenciados, o nome do volume segue o modelo de nome definido no nível raiz na definição do backend.
3. O Trident não suporta o uso de um operador de fatiamento com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, o Trident acrescentará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume econômico do NAS exceder 64 caracteres, 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 nomes, o processo de criação do volume falhará.

Compartilhar um volume NFS entre namespaces

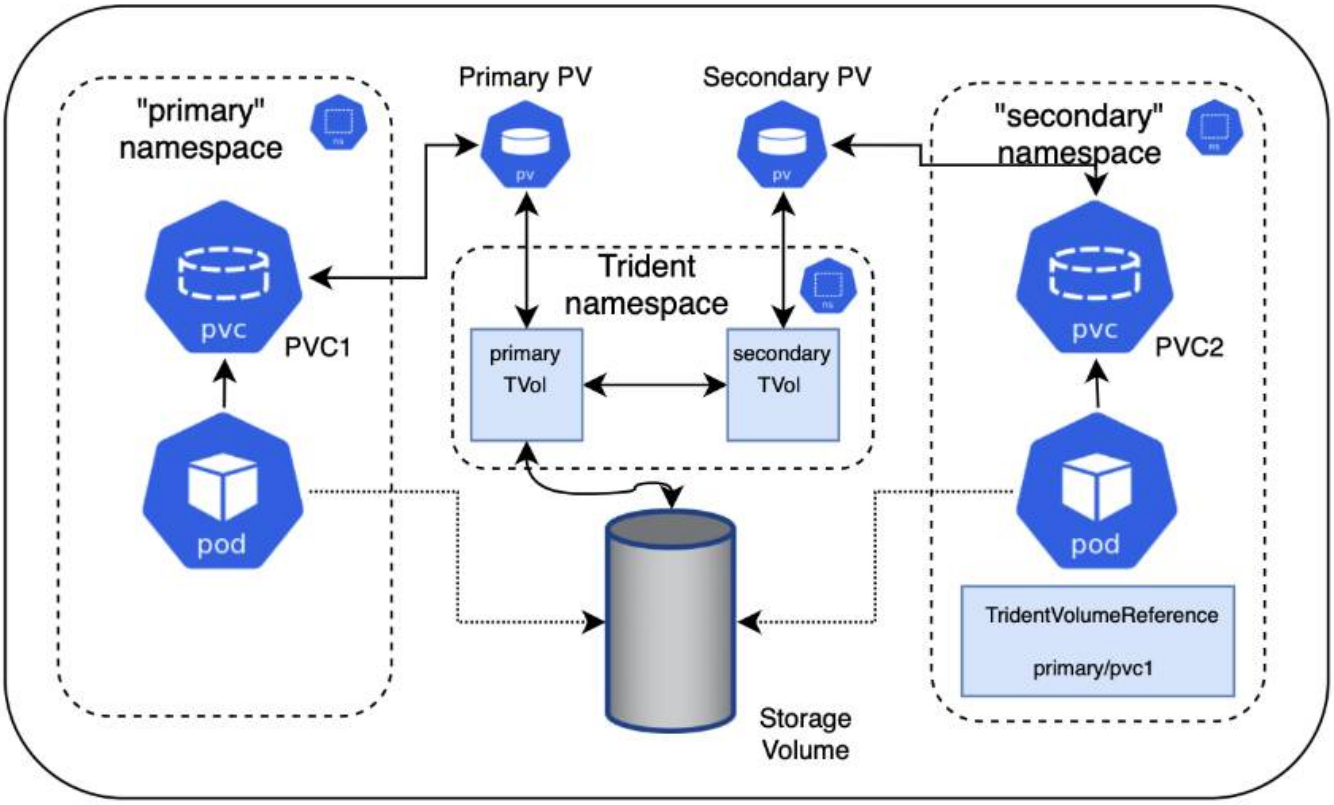
Usando o Trident, você pode criar um volume em um namespace primário e compartilhá-lo em um ou mais namespaces secundários.

Características

O CR TridentVolumeReference permite compartilhar com segurança volumes NFS ReadWriteMany (RWX) em um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes oferece os seguintes benefícios:

- Múltiplos níveis de controle de acesso para garantir a segurança.
- Compatível com todos os drivers de volume Trident NFS.
- Sem dependência do tridentctl ou de qualquer outro recurso não nativo do Kubernetes.

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



Início rápido

Você pode configurar o compartilhamento de volumes NFS em apenas alguns passos.

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 **Conceder permissão para criar uma solicitação de configuração (CR) no espaço de nomes de destino.**

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

3 **Crie uma TridentVolumeReference no namespace de destino.**

O proprietário do namespace de destino cria o CR TridentVolumeReference 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 namespaces exige colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é definida em cada etapa.

Passos

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

   O Trident cria o PV e seu volume de armazenamento NFS de backend.

   - Você pode compartilhar o PVC com vários namespaces usando uma lista separada 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 o `shareToNamespace` Anotações a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja em vigor 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 uma solicitação de configuração (CR) TridentVolumeReference no namespace de destino que faça referência 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:** Criar um PVC(`pvc2`) no espaço de nomes de destino(`namespace2`) usando o `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 diâmetro do tubo de PVC de destino deve ser menor ou igual ao do tubo de PVC de origem.

Resultados

Trident lê o `shareFromPVC` A anotação no PVC de destino 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 do PV de origem. O PVC e o PV de destino parecem estar conectados normalmente.

Excluir um volume compartilhado

Você pode excluir um volume que é 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.

Usar `tridentctl get` 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 a consulta ao volume de origem subordinado.
- `--subordinateOf string`: Limitar a consulta aos subordinados do volume.

Limitações

- O Trident não pode impedir que os namespaces de destino gravem no volume compartilhado. Você deve usar bloqueio de arquivos ou outros processos para evitar a sobrescrita de dados em volumes compartilhados.
- Não é possível revogar o acesso ao PVC de origem removendo o `shareToNamespace` ou `shareFromNamespace` anotações ou exclusão das `TridentVolumeReference` CR. Para revogar o acesso, você deve excluir o PVC subordinado.
- Não é possível criar snapshots, clones ou espelhar volumes subordinados.

Para maiores informações

Para saber mais sobre acesso a volumes entre namespaces:

- Visita"Compartilhamento de volumes entre namespaces: Dê as boas-vindas ao acesso a volumes entre namespaces." .
- Veja a demonstração em"NetAppTV" .

Clonar volumes entre namespaces

Usando o Trident, você pode criar novos volumes utilizando volumes existentes ou snapshots de volumes de um namespace diferente dentro do mesmo cluster Kubernetes.

Pré-requisitos

Antes de clonar volumes, certifique-se de que os backends de origem e destino sejam do mesmo tipo e tenham a mesma classe de armazenamento.

> A clonagem entre namespaces é suportada apenas para o `ontap-san` e `ontap-nas` drivers de armazenamento. Clones somente leitura não são suportados.

Início rápido

Você pode configurar a clonagem de volumes em apenas alguns passos.

Configure o PVC de origem para clonar o volume.

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

Conceder permissão para criar uma solicitação de configuração (CR) no espaço de nomes de destino.

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

Crie uma TridentVolumeReference no namespace de destino.

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

Crie o PVC clonado no namespace de destino.

O proprietário do namespace de destino cria um PVC para clonar o PVC do namespace de origem.

Configure os namespaces de origem e destino.

Para garantir a segurança, a clonagem de volumes entre namespaces requer a colaboração e a ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é definida em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Criar o PVC(`pvc1`) no espaço de nomes de origem(`namespace1`) que concede permissão para compartilhar com o espaço de nomes de destino(`namespace2`) usando o `cloneToNamespace` anotação.

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

A Trident cria o PV e seu volume de armazenamento de backend.

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

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja implementado para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino.(`namespace2`).
3. **Proprietário do namespace de destino:** Crie uma solicitação de configuração (CR) TridentVolumeReference no namespace de destino que faça referência 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:** Criar um PVC(`pvc2`) no espaço de nomes de destino(`namespace2`) usando o `cloneFromPVC` ou `cloneFromSnapshot` , e `cloneFromNamespace` Anotações para designar o PVC de origem.

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

Limitações

- Para PVCs provisionadas usando drivers ontap-nas-economy, clones somente leitura não são suportados.

Replicar volumes usando o SnapMirror

O Trident suporta relações de espelhamento entre um volume de origem em um cluster e o volume de destino no cluster emparelhado para replicação de dados em casos de recuperação de desastres. Você pode usar uma Definição de Recurso Personalizado (CRD) com namespace, chamada Trident Mirror Relationship (TMR), para executar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remover relações de espelhamento entre volumes
- Quebre os relacionamentos espelhados
- Promover o volume secundário em situações de desastre (falhas).
- Realizar a transição sem perda de dados de aplicações de um cluster para outro (durante failovers ou migrações planejadas).

Pré-requisitos para replicação

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

Clusters ONTAP

- * Trident*: A versão 22.10 ou posterior do Trident deve estar presente nos clusters Kubernetes de origem e destino que utilizam o ONTAP como backend.
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror que utilizam o pacote Data Protection devem estar habilitadas nos clusters ONTAP de origem e destino. Consulte "Visão geral do licenciamento do SnapMirror no ONTAP" para mais informações.

  A partir do ONTAP 9.10.1, todas as licenças são fornecidas como um arquivo de licença NetApp (NLF), que é um único arquivo que habilita vários recursos. Consulte"Licenças incluídas no ONTAP One" para mais informações.

  Somente a proteção assíncrona SnapMirror é suportada.

Espionagem

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar interligados. Consulte "Visão geral do peering de clusters e SVMs" para mais informações.

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

- * Trident e SVM*: Os SVMs remotos emparelhados devem estar disponíveis para o Trident no cluster de destino.

Motoristas com suporte

O NetApp Trident oferece suporte à replicação de volumes com a tecnologia NetApp SnapMirror, utilizando

classes de armazenamento com suporte dos seguintes drivers: **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` : FC** `ontap-san` : NVMe/TCP (requer versão mínima do ONTAP 9.15.1)

> A replicação de volumes usando o SnapMirror não é compatível com sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte"Saiba mais sobre os sistemas de armazenamento ASA r2" .

Crie um PVC espelhado

Siga estes passos e utilize os exemplos de CRD para criar uma relação de espelhamento entre os volumes primário e secundário.

Passos

1. Execute os seguintes passos no cluster Kubernetes primário:
   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 uma StorageClass criada anteriormente.

      Exemplo

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

   c. Crie uma relação de espelhamento (MirrorRelationship) 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 do volume e o estado atual de proteção de dados (DP) do volume e, em seguida, preenche o campo de status do MirrorRelationship.

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

```
kubectl 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 os seguintes passos no cluster Kubernetes secundário:

   a. Crie uma StorageClass com o parâmetro trident.netapp.io/replication: 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 relacionamento de espelhamento (MirrorRelationship) 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 configurada (ou padrão para ONTAP) e o inicializará.

c. Crie um PVC com uma StorageClass previamente criada para atuar como destino secundário (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 a existência do CRD TridentMirrorRelationship e não conseguirá criar o volume caso o relacionamento não exista. Caso exista a relação, o Trident garantirá que o novo FlexVol volume seja colocado em uma SVM que esteja emparelhada com a SVM remota definida no MirrorRelationship.

Estados de replicação de volume

Uma relação de espelhamento Trident (TMR, do inglês Trident Mirror Relationship) é um CRD (Componente de Referência Completa) que representa uma das extremidades de uma relação de replicação entre PVCs (Componentes Variáveis Parciais). O TMR de destino possui um estado, que informa ao Trident qual é o estado desejado. O TMR de destino possui os seguintes estados:

- **Estabelecido**: o PVC local é o volume de destino de uma relação de espelhamento, e esta é uma nova relação.
- **Em destaque**: o PVC local é ReadWrite e pode ser montado, sem nenhuma relação de espelhamento em vigor no momento.
- **Reestabelecido**: o PVC local é o volume de destino de uma relação de espelhamento e também estava anteriormente nessa relação de espelhamento.
  - O estado restabelecido deve ser usado se o volume de destino já teve alguma relação com o volume de origem, pois ele sobrescreve o conteúdo do volume de destino.
  - O estado restabelecido falhará se o volume não estiver previamente relacionado com a fonte.

Promover o PVC secundário durante uma falha não planejada.

Execute a seguinte etapa no cluster Kubernetes secundário:

- Atualize o campo *spec.state* de TridentMirrorRelationship para `promoted` .

Promover o PVC secundário durante uma falha planejada

Durante uma migração (failover) planejada, execute os seguintes passos para promover o PVC secundário:

Passos

1. No cluster Kubernetes primário, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster Kubernetes primário, crie o CR SnapshotInfo 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 Kubernetes secundário, atualize o campo *spec.state* do CR *TridentMirrorRelationship* para *promoted* e o campo *spec.promotedSnapshotHandle* para o nome interno do snapshot.
4. No cluster Kubernetes secundário, confirme o status (campo status.state) do TridentMirrorRelationship

como promovido.

Restaurar uma relação de espelhamento após uma falha.

Antes de restaurar a relação de espelhamento, escolha o lado que deseja definir como o novo lado principal.

Passos

1. No cluster Kubernetes secundário, verifique se os valores do campo *spec.remoteVolumeHandle* no TridentMirrorRelationship estão atualizados.
2. No cluster Kubernetes secundário, atualize o campo *spec.mirror* do TridentMirrorRelationship para `reestablished`.

Operações adicionais

O Trident suporta as seguintes operações nos volumes primário e secundário:

Replicar o PVC primário em um novo PVC secundário.

Certifique-se de já possuir um tubo de PVC primário e um tubo de PVC secundário.

Passos

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

Redimensionar um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado normalmente; o ONTAP expandirá automaticamente qualquer flexbox de destino se a quantidade de dados exceder o tamanho atual.

Remover 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 rompe a relação de replicação.
- Ou, atualize o campo spec.state para *promoted*.

Excluir um PVC (que foi previamente espelhado)

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

Excluir um TMR

A exclusão de um TMR em um dos lados 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 haverá relação de espelhamento existente e o TMR será removido, e o Trident promoverá o PVC local para *Leitura/Gravação*. Essa exclusão libera os metadados do SnapMirror para o volume local no ONTAP. Caso este volume seja utilizado em uma relação de espelhamento no futuro, será necessário usar um novo TMR com um estado de replicação de volume *estabelecido* ao criar a nova relação de espelhamento.

Atualize os relacionamentos de espelhamento quando o ONTAP estiver online.

As relações de espelhamento podem ser atualizadas a qualquer momento após serem estabelecidas. Você pode usar o `state: promoted` ou `state: reestablished` campos para atualizar os relacionamentos. 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 espelhamento 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 ONTAP . Consulte o seguinte exemplo de formato 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 TridentActionMirrorUpdate. Pode assumir um valor entre *Succeeded*, *In Progress* ou *Failed*.

Utilizar a topologia CSI

O Trident pode criar e anexar volumes seletivamente aos nós presentes em um cluster Kubernetes, utilizando o "Recurso de topologia CSI" .

Visão geral

Usando o recurso Topologia CSI, o acesso aos volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Atualmente, os provedores de nuvem permitem que os administradores do Kubernetes criem nós baseados em zonas. Os nós podem estar localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para cargas de trabalho em uma arquitetura multizona, o Trident usa a topologia CSI.

Saiba mais sobre o recurso de Topologia CSI "aqui" .

O Kubernetes oferece dois modos exclusivos de vinculação de volumes:

- Com `VolumeBindingMode` definido para `Immediate` O Trident cria o volume sem qualquer conhecimento de topologia. A vinculação de volumes e o provisionamento dinâmico são tratados quando o PVC é criado. Este é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Volumes persistentes são criados sem qualquer dependência dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido para `WaitForFirstConsumer` A criação e vinculação de um Volume Persistente para um PVC é adiada até que um pod que utilize o PVC seja agendado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos

requisitos de topologia.

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

O que você vai precisar

Para utilizar a topologia CSI, você precisa do seguinte:

- Um cluster Kubernetes executando um"versão do Kubernetes suportada"

```
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 do cluster devem ter rótulos que introduzam informações sobre a topologia.(`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Essas etiquetas **devem estar presentes nos nós do cluster** antes da instalação do Trident para que ele reconheça a 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","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Passo 1: Crie um backend com reconhecimento de topologia

Os sistemas de armazenamento Trident podem ser configurados para provisionar volumes seletivamente com base em zonas de disponibilidade. Cada backend pode conter um opcional `supportedTopologies` Bloco que representa uma lista de zonas e regiões suportadas. Para StorageClasses que fazem uso desse backend, 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

```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

```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 utilizado 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 uma StorageClass. Para StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um backend, o Trident cria um volume no backend.

Você pode definir `supportedTopologies` por pool de armazenamento também. 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, o `region` e `zone` As etiquetas indicam a localização da piscina de armazenamento. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` ditam de onde os pools de armazenamento podem ser consumidos.

Etapa 2: Defina StorageClasses que levem em consideração a topologia.

Com base nos rótulos de topologia fornecidos aos nós do cluster, as StorageClasses podem ser definidas para conter informações de topologia. Isso determinará os pools de armazenamento que servem como candidatos para solicitações de PVC feitas e o subconjunto de nós que podem usar os volumes provisionados pelo Trident.

Veja o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - 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 de StorageClass fornecida acima, `volumeBindingMode` está definido para `WaitForFirstConsumer` . Os PVCs solicitados com esta StorageClass não serão processados até que sejam referenciados em um pod. E, `allowedTopologies` Fornece as zonas e a região a serem utilizadas. O `netapp-san-us-east1` O StorageClass cria PVCs no `san-backend-us-east1` backend definido acima.

Etapa 3: Criar e usar um PVC

Com a StorageClass criada e mapeada para um backend, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

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

A criação de um PVC usando este manifesto resultaria no seguinte:

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

Para que o Trident crie um volume e o fixe ao PVC, utilize o PVC em um invólucro. 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 que estejam presentes no `us-east1` região, e escolha qualquer nó que esteja presente na `us-east1-a` ou `us-east1-b` zonas.

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
```

Atualizar os backends para incluir `supportedTopologies`

Os backends preexistentes podem ser atualizados para incluir uma lista de `supportedTopologies` usando `tridentctl backend update` . Isso não afetará os volumes que já foram provisionados e será usado apenas para PVCs subsequentes.

Encontre mais informações

- "Gerenciar recursos para contêineres"
- "seletor de nó"
- "Afinidade e antiafinidade"
- "Manchas e Tolerâncias"

Trabalhar com instantâneos

Os snapshots de volumes persistentes (PVs) do Kubernetes permitem a criação de cópias pontuais dos 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 volumes a partir de snapshots.

Visão geral

O snapshot de volume é suportado por `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `gcp-cvs` , `azure-netapp-files` , e `google-cloud-netapp-volumes` motoristas.

Antes de começar

Para trabalhar com snapshots, você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs). Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a sua distribuição Kubernetes não incluir o controlador de snapshots e os CRDs, consulteImplante um controlador de instantâneo de volume .

Não crie um controlador de snapshots se estiver criando snapshots de volume sob demanda em um ambiente GKE. O GKE utiliza um controlador de snapshots integrado e oculto.

Criar um instantâneo de volume

Passos

1. Criar um `VolumeSnapshotClass` Para obter mais informações, consulte"VolumeSnapshotClass" .
   - O `driver` Aponta para o motorista do Trident CSI.
   - `deletionPolicy`pode ser `Delete` ou `Retain` . Quando definido para `Retain` , o snapshot físico subjacente no cluster de armazenamento é mantido mesmo quando o `VolumeSnapshot` O objeto foi 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 uma imagem instantânea de um tubo de 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 de instantâneo de volume para um PVC chamado `pvc1` e o nome do instantâneo está definido como `pvc1-snap` . Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa a captura instantânea real.

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

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

- Você pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` O VolumeSnapshot é descrito dessa forma. O `Snapshot Content Name` Identifica o objeto VolumeSnapshotContent que fornece este snapshot. O `Ready To Use` O 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
...
```

Criar um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um VolumeSnapshot chamado `<pvc-name>` como fonte dos dados. Após a fabricação do PVC, ele pode ser fixado a um invólucro e utilizado como qualquer outro PVC.

> O PVC será criado no mesmo backend que o volume de origem. Consulte"KB: A criação de um PVC a partir de um Snapshot de PVC do Trident não pode ser feita em um backend alternativo." .

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: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importar um instantâneo de volume

A Trident apoia o"Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador do cluster crie um `VolumeSnapshotContent` snapshots de objetos e importações criados fora do Trident.

Antes de começar

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

Passos

1. **Administrador do cluster:** Criar um `VolumeSnapshotContent` objeto que faz referência ao snapshot do backend. Isso inicia o fluxo de trabalho de captura de instantâneos no Trident.
   - Especifique o nome do snapshot do backend em `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especificar `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle` Esta é a única informação fornecida ao Trident pelo snapshotter externo. `ListSnapshots` chamar.

     > O `<volumeSnapshotContentName>` Nem sempre é possível que o nome do snapshot do backend corresponda ao nome original devido a restrições de nomenclatura do CR.

     Exemplo

     O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência a um snapshot do backend `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. **Administrador do cluster:** Crie o `VolumeSnapshot` CR que faz referência ao `VolumeSnapshotContent` objeto. Este pedido permite o acesso para usar o `VolumeSnapshot` em um determinado espaço de nomes.

   Exemplo

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

   ```
   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 o `ListSnapshots` chamar. Trident cria o `TridentSnapshot` .
   - O capturador de instantâneos externo define o `VolumeSnapshotContent` para `readyToUse` e o `VolumeSnapshot` para `true` .
   - Trident retorna `readyToUse=true` .
4. **Qualquer usuário:** Criar um `PersistentVolumeClaim` para fazer referência ao novo `VolumeSnapshot` , onde o `spec.dataSource` (ou `spec.dataSourceRef`) o nome é o `VolumeSnapshot` nome.

Exemplo

O exemplo a seguir cria um PVC que faz referência a `VolumeSnapshot` nomeado `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
```

Recupere dados de volume usando snapshots

O diretório de snapshots fica oculto por padrão para facilitar a máxima compatibilidade dos volumes provisionados usando o `ontap-nas` e `ontap-nas-economy` motoristas. Ative o `.snapshot` diretório para recuperar dados diretamente de snapshots.

Utilize o comando `volume snapshot restore` da CLI do ONTAP para restaurar um volume a um estado registrado em um snapshot anterior.

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

Ao restaurar uma cópia de snapshot, a configuração de volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia instantânea serão perdidas.

Restauração de volume in-place a partir de um snapshot

O Trident proporciona restauração de volume rápida e in situ a partir de uma imagem instantânea usando o `TridentActionSnapshotRestore` (TASR) CR. Essa solicitação de configuração (CR) funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

O Trident suporta a restauração de snapshots no `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` , `google-cloud-netapp-volumes` , e `solidfire-san` motoristas.

Antes de começar

Você precisa ter um PVC encadernado e um snapshot de volume disponível.

- Verifique se o status do PVC está vinculado.

```
kubectl get pvc
```

- Verifique se o snapshot do volume está pronto para uso.

```
kubectl get vs
```

Passos

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

   O TASR CR deve estar em um namespace onde o PVC e o 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
```

2. Aplique a solicitação de restauração (CR) para restaurar a partir do snapshot. Este exemplo restaura a partir de um snapshot. `pvc1` .

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

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

Resultados

O Trident restaura os dados a partir do snapshot. Você pode verificar o status da restauração do 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 novamente a operação automaticamente em caso de falha. Você precisará repetir a operação.
- Usuários do Kubernetes sem acesso de administrador podem precisar de permissão do administrador para criar um CR do TASR em seu namespace de aplicação.

Excluir um PV com snapshots associados

Ao excluir um Volume Persistente com snapshots associados, o volume Trident correspondente é atualizado para o estado "Excluindo". Remova os snapshots de volume para excluir o volume Trident .

Implante um controlador de instantâneo de volume

Se a sua distribuição Kubernetes não incluir o controlador de snapshots e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de snapshot 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.yam
l
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 de 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 atualizar `namespace` para o seu espaço de nomes.

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Trabalhar com snapshots de grupos de volumes

O NetApp Trident permite criar snapshots de vários volumes (um grupo de snapshots de volumes) em grupos de volumes do Kubernetes. Este instantâneo do grupo de volumes representa cópias de vários volumes feitas no mesmo momento.

VolumeGroupSnapshot é um recurso beta do Kubernetes com APIs beta. O Kubernetes 1.32 é a versão mínima necessária para o VolumeGroupSnapshot.

Criar snapshots de grupos de volumes

O snapshot do grupo de volumes é compatível com o `ontap-san` Driver compatível apenas com o protocolo iSCSI, ainda não suportado com Fibre Channel (FCP) nem NVMe/TCP. Antes de começar

- Certifique-se de que sua versão do Kubernetes seja K8s 1.32 ou superior.
- Para trabalhar com snapshots, você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs). Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

  Se a sua distribuição Kubernetes não incluir o controlador de snapshots externo e os CRDs, consulteImplante um controlador de instantâneo de volume .

  > Não crie um controlador de snapshots se estiver criando snapshots de grupos de volumes sob demanda em um ambiente GKE. O GKE utiliza um controlador de snapshots integrado e oculto.

- No arquivo YAML do controlador de snapshots, defina o `CSIVolumeGroupSnapshot` Defina o recurso como 'true' para garantir que o snapshot do grupo de volumes esteja habilitado.
- Crie as classes de snapshot de grupo de volumes necessárias antes de criar um snapshot de grupo de volumes.
- Certifique-se de que todos os PVCs/volumes estejam no mesmo SVM para poder criar o VolumeGroupSnapshot.

Passos

- Crie uma classe VolumeGroupSnapshotClass antes de criar um VolumeGroupSnapshot. Para obter mais informações, consulte"VolumeGroupSnapshotClass" .

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Crie PVCs com os rótulos necessários usando as classes de armazenamento existentes ou adicione esses rótulos aos PVCs existentes.

  O exemplo a seguir cria o PVC usando `pvc1-group-snap` como fonte de dados e rótulo `consistentGroupSnapshot: groupA` . Defina a chave e o valor do rótulo com base em suas necessidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crie um VolumeGroupSnapshot com o mesmo rótulo.(`consistentGroupSnapshot: groupA`) especificado no PVC.

  Este exemplo cria um instantâneo de um grupo de volumes:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recupere dados de volume usando um instantâneo de grupo.

Você pode restaurar Volumes Persistentes individuais usando os snapshots individuais que foram criados como parte do Snapshot do Grupo de Volumes. Não é possível recuperar o Snapshot do Grupo de Volumes como uma unidade.

Utilize o comando `volume snapshot restore` da CLI do ONTAP para restaurar um volume a um estado registrado em um snapshot anterior.

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

Ao restaurar uma cópia de snapshot, a configuração de volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia instantânea serão perdidas.

Restauração de volume in-place a partir de um snapshot

O Trident proporciona restauração de volume rápida e in situ a partir de uma imagem instantânea usando o `TridentActionSnapshotRestore` (TASR) CR. Essa solicitação de configuração (CR) funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Para mais informações, consulte "Restauração de volume in-place a partir de um snapshot".

Excluir um PV com snapshots de grupo associados

Ao excluir um instantâneo de volume de grupo:

- Você pode excluir o grupo de snapshots VolumeGroupSnapshots como um todo, e não snapshots individuais dentro do grupo.
- Se os PersistentVolumes forem excluídos enquanto existir um snapshot para esse PersistentVolume, o Trident moverá esse volume para um estado de "exclusão", pois o snapshot deve ser removido antes que o volume possa ser removido com segurança.
- Se um clone tiver sido criado usando um snapshot agrupado e, em seguida, o grupo for excluído, uma operação de divisão no clone será iniciada e o grupo não poderá ser excluído até que a divisão seja concluída.

Implante um controlador de instantâneo de volume

Se a sua distribuição Kubernetes não incluir o controlador de snapshots e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de snapshot de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crie o controlador de instantâneo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualizar `namespace` para o seu espaço de nomes.

Links relacionados

- "VolumeGroupSnapshotClass"
- "Instantâneos de volume"

Informações sobre direitos autorais

Informações sobre marcas comerciais