Provisionar e gerenciar volumes

Trident

NetApp
July 01, 2026

Índice

Provisionar e gerenciar volumes . . . 1
- Provisionar um volume . . . 1
  - Visão geral . . . 1
  - Crie o PVC . . . 1
- Expandir volumes . . . 4
  - Expandir um volume iSCSI . . . 4
  - Expandir um volume FC . . . 8
  - Expandir um volume NFS . . . 12
- Entenda os limites do subsistema RWX NVMe . . . 15
  - Entenda o limite de 64 nós . . . 15
  - Entenda os modelos de subsistema NVMe . . . 15
  - Identifique sintomas de erro . . . 16
  - Resolver erros de limite do subsistema . . . 16
  - Atualize Trident para aplicar o modelo de super-subsistema . . . 16
- Escalabilidade do controlador . . . 17
  - Conceitos e definições principais . . . 17
  - Suporte à escalabilidade do controlador . . . 17
  - Habilitar escalabilidade do controlador . . . 17
  - Comportamento de concorrência . . . 20
  - Limitações e considerações conhecidas . . . 20
  - Ressalvas e limitações . . . 20
  - Recomendações . . . 20
- Expansão de volume automática . . . 21
  - Requisitos . . . 21
  - Limitações . . . 21
  - Provisionar volumes com política de crescimento automático . . . 22
  - Criar uma política de crescimento automático . . . 22
  - Criar uma política de crescimento automático . . . 23
  - Estados de política . . . 23
  - Associe uma política a um StorageClass . . . 24
  - Precedência de política . . . 24
  - Exemplos de configuração . . . 25
- Gerenciar políticas de crescimento automático . . . 28
  - Ver políticas de Autogrow . . . 28
  - Atualizar uma política de crescimento automático . . . 28
  - Excluir uma política de crescimento automático . . . 29
  - Monitorar o uso da política Autogrow . . . 30
  - Protocolos suportados . . . 30
  - Limitações conhecidas . . . 31
  - Perguntas frequentes . . . 31
- Importar volumes . . . 37
  - Visão geral e considerações . . . 37
  - Importar um volume . . . 38

Exemplos . . . 40
Exemplos de SAN-economy do ONTAP . . . 45
Personalize os nomes e rótulos dos volumes . . . 49
Antes de começar . . . 49
Limitações . . . 49
Principais comportamentos dos nomes de volume personalizáveis . . . 50
Exemplos de configuração de backend com template de nome e rótulos . . . 50
Exemplos de modelos de nome . . . 51
Pontos a considerar . . . 52
Compartilhe um volume NFS entre namespaces . . . 52
Características . . . 52
Início rápido . . . 53
Configure os namespaces de origem e destino . . . 54
Excluir um volume compartilhado . . . 55
Use `tridentctl get` para consultar volumes subordinados . . . 55
Limitações . . . 56
Para obter mais informações . . . 56
Clonar volumes entre namespaces . . . 56
Pré-requisitos . . . 56
Início rápido . . . 56
Configure os namespaces de origem e destino . . . 57
Limitações . . . 59
Replicar volumes usando SnapMirror . . . 59
Pré-requisitos de replicação . . . 59
Crie um PVC espelhado . . . 60
Estados de replicação de volume . . . 63
Promover o PVC secundário durante uma falha não planejada . . . 63
Promover o PVC secundário durante uma falha planejada . . . 63
Restaurar uma relação de espelhamento após uma falha . . . 64
Operações adicionais . . . 64
Atualize os relacionamentos de espelhamento quando ONTAP estiver online . . . 65
Atualizar relações de espelhamento quando ONTAP estiver offline . . . 65
Usar a topologia CSI . . . 65
Visão geral . . . 65
Etapa 1: crie um backend com reconhecimento de topologia . . . 67
Etapa 2: Defina StorageClasses que são cientes da topologia . . . 69
Etapa 3: criar e usar um PVC . . . 70
Atualize os backends para incluir `supportedTopologies` . . . 73
Encontre mais informações . . . 73
Trabalhar com Snapshots . . . 73
Visão geral . . . 73
Criar um snapshot de volume . . . 74
Crie um PVC a partir de um instantâneo de volume . . . 75
Importar um Snapshot de volume . . . 76
Recupere dados de volume usando Snapshots . . . 78

Restauração de volume in-place a partir de uma Snapshot . . . 78
Excluir um PV com snapshots associados . . . 80
Implante um controlador de Snapshot de volume . . . 80
Links relacionados . . . 81
Trabalhar com Snapshots de grupo de volume . . . 81
Criar snapshots de grupo de volume . . . 82
Recuperar dados de volume usando um snapshot de grupo . . . 83
Restauração de volume in-place a partir de uma Snapshot . . . 84
Excluir um PV com snapshots de grupo associados . . . 84
Implante um controlador de Snapshot de volume . . . 84
Links relacionados . . . 85

Provisionar e gerenciar volumes

Provisionar um volume

Crie um PersistentVolumeClaim (PVC) que utiliza o StorageClass do Kubernetes configurado para solicitar acesso ao PV. Em seguida, você pode montar o PV em um pod.

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. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o PVC, você pode montar o volume em um pod.

Crie 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. Monte 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

Manifestos 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 um StorageClass chamado `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 que está associado a um StorageClass chamado `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 anexar 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 os parâmetros para controlar como Trident provisiona volumes.

Expandir volumes

Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes após eles serem 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 iSCSI Persistent Volume (PV) usando o provisionador CSI.

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

Passo 1: Configurar o StorageClass para suportar a expansão de volume

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

```
cat storageclass-ontapsan.yaml
```

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

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

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o 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
```

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (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 pod que conecte o PVC

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

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da operação de expansão.

Neste exemplo, um pod é criado que usa o `san-pvc`.

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

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

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 validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do 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 FC Persistent Volume (PV) usando o provisionador CSI.

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

Passo 1: Configurar o StorageClass para suportar a expansão de volume

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

```
cat storageclass-ontapsan.yaml
```

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

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

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o 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
```

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (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 pod que conecte o PVC

Conecte o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um PV FC:

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da

operação de expansão.

Neste exemplo, um pod é criado que usa o `san-pvc`.

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

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

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 validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do 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

Trident suporta expansão de volume para NFS PVs provisionados em `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, e `azure-netapp-files` backends.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

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

Passo 2: Crie um PVC com o StorageClass que você criou

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

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

Trident deve criar um NFS PV 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 Trident volume:

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

Entenda os limites do subsistema RWX NVMe

ReadWriteMany (RWX) volumes que utilizam o protocolo NVMe têm um limite de escalabilidade de 64 nós por volume. A seguir, são apresentadas as limitações, explicada a arquitetura do subsistema NVMe envolvido e descritas as etapas necessárias para a resolução do problema.

Entenda o limite de 64 nós

Se você planeja usar ReadWriteMany (RWX) volumes com o protocolo NVMe, um único volume RWX NVMe não pode ser montado por mais de 64 nós em um cluster Kubernetes.

Não agende cargas de trabalho que montem o mesmo RWX NVMe PersistentVolumeClaim em mais de 64 nós.

Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.

Entenda os modelos de subsistema NVMe

Modelo de subsistema por volume (Trident releases anteriores a 26.02)

Nas versões do Trident anteriores à 26.02, os volumes RWX NVMe são provisionados usando um modelo de subsistema por volume. Cada volume RWX NVMe é mapeado para seu próprio subsistema NVMe dedicado no ONTAP.

Este modelo é simples, mas possui um limite de escalabilidade inferior. Em clusters Kubernetes de grande porte, os limites dos controladores de subsistema são atingidos rapidamente porque cada volume RWX consome um subsistema dedicado.

Modelo de super-subsistema (introduzido no Trident 26.02)

A partir do Trident 26.02, os volumes RWX NVMe utilizam um modelo de super-subsistema compartilhado. Vários volumes RWX NVMe compartilham o mesmo subsistema NVMe.

Cada super-subsistema suporta até 1024 namespaces (volumes). Esse modelo melhora significativamente a escalabilidade para cargas de trabalho RWX e reduz a probabilidade de atingir os limites do subsistema ONTAP.

Cada volume RWX NVMe suporta até 64 nós.

Identifique sintomas de erro

Se você criar ou anexar volumes RWX NVMe em grande escala, poderá observar erros semelhantes aos seguintes:

```
Maximum number of controllers reached. No more controllers can be created.
```

Este erro indica que o limite do controlador do subsistema NVMe do ONTAP foi atingido.

Resolver erros de limite do subsistema

Para superar as limitações de subsistemas por volume e aproveitar as vantagens do modelo de super-subsistema, atualize para Trident 26.02 ou posterior.

Atualize Trident para aplicar o modelo de super-subsistema

Para aplicar o modelo de super-subsistema para volumes RWX NVMe:

1. Atualize Trident para a versão 26.02 ou posterior.
2. Reduza para zero réplicas todos os pods que usam volumes RWX NVMe.
3. Verifique se nenhuma carga de trabalho está usando ativamente volumes RWX NVMe.
4. Aumente a escala dos pods novamente.

Essa sequência de reinicialização garante que os volumes RWX NVMe sejam conectados usando o modelo de super-subsistema.

- Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.
- O limite de 64 nós se aplica por volume RWX NVMe.
- Outros modos de acesso e outros protocolos não são afetados.

Escalabilidade do controlador

Trident introduz escalabilidade do controlador por meio de maior simultaneidade entre vários drivers de armazenamento. Os clientes podem identificar quais drivers do Trident oferecem suporte à escalabilidade do controlador na disponibilidade geral e quais drivers estão disponíveis como prévia técnica no Trident 26.02. Isso permite decisões de implantação informadas e o gerenciamento adequado de riscos para ambientes Kubernetes escaláveis.

Conceitos e definições principais

Escalabilidade do controlador

A escalabilidade do controlador refere-se à capacidade do controlador Trident de processar múltiplas operações de armazenamento em paralelo, em vez de serializá-las por trás de um único bloqueio. Essas operações incluem criação, exclusão e redimensionamento de volumes, criação e exclusão de snapshots, publicação e cancelamento de publicação de volumes e gerenciamento de backend.

Quando a escalabilidade do controlador está habilitada, as operações em diferentes volumes e backends são executadas simultaneamente. Isso aumenta a taxa de transferência e reduz completamente o tempo de operação em ambientes com um grande número de operações simultâneas de PersistentVolumeClaim e VolumeSnapshot.

Suporte à escalabilidade do controlador

Trident oferece suporte à escalabilidade com diferentes níveis de maturidade, dependendo do driver de armazenamento.

Disponibilidade geral

Os seguintes drivers oferecem suporte à escalabilidade em disponibilidade geral no Trident 26.02:

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

> Os `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` drivers são diferentes. Apenas `google-cloud-netapp-volumes` é suportado. Não use `google-cloud-netapp-volumes-san` em configurações de backend ou exemplos.

Habilitar escalabilidade do controlador

A escalabilidade do controlador é controlada pela `enableConcurrency` opção de configuração. Essa opção deve ser explicitamente habilitada durante a instalação do Trident ou atualizando uma implantação existente.

Desdobramento do operador Trident

Para habilitar a escalabilidade do controlador com o Trident operator, defina `enableConcurrency` como `true` no `TridentOrchestrator` recurso personalizado.

Nova instalação

Crie ou edite o `TridentOrchestrator` CR com `enableConcurrency` definido como `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Aplique o CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Instalação existente

Corrija a `TridentOrchestrator`CR existente para permitir a escalabilidade do controlador:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Verifique se a configuração foi aplicada:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Implantação Helm

Para habilitar a escalabilidade do controlador com o Helm, defina o valor `enableConcurrency` como `true`.

Nova instalação

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Instalação existente

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Alternativamente, defina `enableConcurrency` para `true` em um arquivo `values.yaml` personalizado:

```
# values.yaml
enableConcurrency: true
```

Em seguida, instale ou atualize usando o arquivo de valores:

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

implantação do tridentctl

Para habilitar a escalabilidade do controlador com `tridentctl`, passe a flag `--enable-concurrency` durante a instalação.

Nova instalação

```
tridentctl install -n trident --enable-concurrency
```

Instalação existente

Para habilitar a escalabilidade do controlador em uma implantação existente baseada em `tridentctl`, desinstale e reinstale com a flag:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Verifique se a escalabilidade está ativada

Após habilitar a escalabilidade do controlador, verifique se o Trident controller está em execução com a concorrência habilitada consultando os logs do pod do controlador:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Você deverá ver uma entrada de log indicando que a simultaneidade está habilitada.

Prévia técnica

Os seguintes drivers oferecem suporte à escalabilidade como uma prévia técnica no Trident 26.02:

- `nas-eco`
- `san-eco`

Para estes drivers:

- A concorrência de controladores está disponível para avaliação e teste
- O comportamento pode mudar em versões futuras
- O uso em ambientes de produção não é recomendado

Comportamento de concorrência

Quando a escalabilidade do controlador está ativada:

- Trident substitui o bloqueio global único por bloqueios granulares, por recurso
- Operações que modificam o mesmo recurso são serializadas para manter a consistência de dados
- Operações que apenas leem de um recurso podem prosseguir simultaneamente com outras operações de leitura nesse recurso
- Trident limita as solicitações simultâneas da API ONTAP a 20 por LIF de gerenciamento para evitar sobrecarga dos sistemas de armazenamento de backend
- Se vários backends compartilharem a mesma LIF de gerenciamento, eles compartilham esse limite de 20 solicitações

Limitações e considerações conhecidas

As seguintes considerações aplicam-se à escalabilidade do controlador:

- A concorrência é gerenciada internamente pelo controlador Trident
- Não há limites de simultaneidade configuráveis pelo usuário nesta versão
- A produtividade geral depende de:
  - O driver de armazenamento em uso
  - Responsividade do backend
  - Desempenho do servidor da API do Kubernetes
- Alta concorrência pode aumentar a carga nos sistemas de armazenamento de backend

Ressalvas e limitações

As seguintes limitações se aplicam em Trident 26.02:

- O comportamento de escalabilidade do controlador não é idêntico em todos os drivers
- Os drivers de pré-visualização técnica podem apresentar:
  - Desempenho inconsistente sob alta carga
  - Alterações de comportamento entre versões
- A depuração de operações simultâneas pode ser mais complexa devido à execução paralela
- As métricas e os logs podem mostrar saída de operações intercaladas

Recomendações

- Utilize drivers de disponibilidade geral (GA) para ambientes de produção que exigem alta escalabilidade
- Avalie drivers de pré-visualização técnica em ambientes de não produção
- Monitore o desempenho do backend e do controlador ao operar em escala

- Evite presumir a ordem das operações em scripts de automação

Expansão de volume automática

A expansão de volume permite que os Volumes Persistentes provisionados pelo Trident cresçam automaticamente quando a capacidade utilizada atinge um limite definido. Essa funcionalidade reduz a sobrecarga operacional e ajuda a evitar interrupções nos aplicativos causadas pelo esgotamento da capacidade. A expansão de volume automática é implementada usando Autogrow Policies. Uma Autogrow Policy define:

- O limite de utilização que desencadeia a expansão
- A quantidade pela qual o volume cresce
- O tamanho máximo que o volume pode atingir

> Os volumes aumentam de tamanho automaticamente quando o limite de utilização definido é excedido. Os volumes nunca são reduzidos automaticamente.

Requisitos

Antes de configurar a expansão automática de volume, certifique-se de que os seguintes requisitos sejam atendidos:

- Trident 26.02 ou posterior
- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos personalizados
- StorageClasses configurado com `allowVolumeExpansion: true`
- Protocolos ONTAP suportados:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Memória Não Volátil Express (NVMe)
  - Protocolo Fibre Channel (FCP)

Limitações

- Os volumes de bloco bruto do ONTAP Non-Volatile Memory Express anteriores ao ONTAP 9.16.1 não suportam expansão automática.
- Para volumes de rede de área de armazenamento, se `growthAmount` for menor ou igual a 50 mebibytes, Trident aumenta automaticamente o valor para 51 mebibytes antes de redimensionar, desde que o tamanho resultante não exceda `maxSize`.
- Em ambientes já existentes, a expansão automática pode não funcionar para determinados volumes existentes devido ao comportamento de migração de publicação de volume.
- Quando um volume atinge `maxSize`, nenhuma expansão adicional ocorre.
- Protocolos suportados para expansão automática de volume:
  - Network File System (NFS)

- Internet Small Computer Systems Interface (iSCSI)
- Memória Não Volátil Express (NVMe)
- Protocolo Fibre Channel (FCP)

Provisionar volumes com política de crescimento automático

A política de volume com crescimento automático pode ser configurada em dois níveis:

- Nível da classe de armazenamento: define o padrão para todos os volumes (usando anotação)
- Nível PVC: substitui o padrão da classe de armazenamento (usando anotação)

Criar uma política de crescimento automático

As políticas de crescimento automático permitem a expansão automática do volume quando os volumes atingem um limite de capacidade definido.

Certifique-se de ter:

- Trident 26.02 ou posterior instalado
- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos
- Compreensão dos requisitos de crescimento da carga de trabalho

Uma Autogrow Policy define como os volumes se expandem automaticamente quando atingem um limite de capacidade definido.

Você pode criar políticas de crescimento automático em qualquer ponto do seu fluxo de trabalho:

- Antes de StorageClasses e volumes serem criados
- Após StorageClasses existirem
- Após o provisionamento dos volumes

Essa flexibilidade permite que você introduza a expansão automática sem recriar os recursos existentes.

Especificações da política de volume com crescimento automático

As políticas de crescimento automático são recursos personalizados do Kubernetes definidos da seguinte forma:

Campo	Descrição	Formatar	Obrigatório	Exemplo	Padrão
nome	Identificador de política exclusivo	String	Sim	política-db-produção	Nenhum
usedThreshold	Percentual de capacidade que desencadeia a expansão	String de porcentagem	Sim	"80%"	Nenhum
growthAmount	Quantidade a crescer quando o limite for atingido	Porcentagem ou tamanho	Não	"10%" ou "5Gi"	"10%"

Campo	Descrição	Formatar	Obrigatório	Exemplo	Padrão
maxSize	Limite máximo de tamanho do volume	Quantidade de Kubernetes	Não	"500Gi"	Ilimitado

Criar uma política de crescimento automático

Passos

1. Crie um arquivo YAML que defina sua Autogrow Policy:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Aplique a política ao seu cluster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verifique se a política foi criada:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Resultado esperado

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

Estados de política

Depois que você cria uma política, Trident valida a especificação e atribui um dos seguintes estados:

Estado	Descrição	Ação necessária
Sucesso	A política é validada e está pronta para uso.	Nenhum.
Falha	Foram detectados erros de validação.	Revise e corrija a especificação.

Estado	Descrição	Ação necessária
Excluindo	A exclusão está em andamento.	Aguarde a conclusão.

Associe uma política a um StorageClass

Você pode associar uma Autogrow Policy a um StorageClass usando a `trident.netapp.io/autogrowPolicy` anotação. Todos os volumes provisionados a partir desse StorageClass herdam a política.

O StorageClass deve ter `allowVolumeExpansion: true`.

Passos

1. Crie ou modifique uma StorageClass com a anotação de Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Aplique o StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifique a anotação:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
production-db-policy
```

Precedência de política

Quando as anotações da Política de Crescimento Automático são definidas tanto em um StorageClass quanto em um PVC, Trident aplica as seguintes regras de precedência:

1. **A anotação PVC tem prioridade.** Se um PVC definir `trident.netapp.io/autogrowPolicy`, esse valor será sempre usado.
2. **StorageClass a anotação só se aplica quando o PVC não possui nenhuma anotação.**
3. **Se nenhum dos dois tiver a anotação,** nenhuma Autogrow Policy será aplicada.

StorageClass anotação	Anotação PVC	Comportamento eficaz
`trident.netapp.io/autogrowPolicy: standard-agp`	Não definido	Usos `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Usa `logs-policy` (PVC substitui StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Sem política de crescimento automático (PVC desativa o autogrow).
Não definido	`trident.netapp.io/autogrowPolicy: dev-policy`	Usos `dev-policy`.
Não definido	Não definido	Sem política de autogrow.

Exemplos de configuração

Os exemplos a seguir mostram configurações comuns de Autogrow Policy para diferentes casos de uso.

Política conservadora para bancos de dados de produção

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Armazenamento de logs com incrementos de crescimento fixos

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Política mínima com valores padrão

Quando você omite `growthAmount` e `maxSize`, Trident usa os padrões (`10%` de crescimento, tamanho ilimitado):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Política com um maxSize personalizado e growthAmount padrão

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Crescimento agressivo com maxSize ilimitado

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Política com percentuais fracionários

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

São aceitas porcentagens fracionárias. Se você especificar mais de três casas decimais, Trident arredonda o valor para três casas decimais.

NAS StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Gerenciar políticas de crescimento automático

Após criar as Autogrow Policies, você pode visualizá-las, atualizá-las e excluí-las conforme necessário. Você também pode monitorar quais volumes estão usando uma determinada policy.

Ver políticas de Autogrow

Listar todas as políticas

Use `kubectl` para listar todas as Autogrow Policies no seu cluster:

```
kubectl get tridentautogrowpolicy
```

Alternativamente, use `tridentctl`:

```
tridentctl get autogrowpolicy
```

Ver detalhes da política

Para visualizar a especificação completa e o status de uma policy:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Para visualizar uma apólice com seus respectivos volumes em formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Atualizar uma política de crescimento automático

Você pode modificar uma política existente para alterar seu limite, quantidade de crescimento ou tamanho máximo. As alterações entram em vigor imediatamente para todos os volumes que utilizam a política.

> As alterações afetam todos os volumes que atualmente utilizam a política. Teste as alterações primeiro em um ambiente que não seja de produção, sempre que possível.

Passos

1. Edite a política:

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifique os `spec` fields conforme necessário:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. Salve e saia. As alterações entram em vigor imediatamente.

Considerações sobre atualização

- **Efeito imediato:** Todos os volumes que utilizam a política adotam os novos parâmetros na próxima avaliação de crescimento.
- **Não é necessário reiniciar o volume:** As alterações serão aplicadas na próxima operação de crescimento.
- **Teste primeiro:** Valide as alterações em um ambiente que não seja de produção quando possível.
- **Comunique alterações:** Notifique as equipes quando você modificar as políticas compartilhadas.

Excluir uma política de crescimento automático

As políticas de crescimento automático usam proteção de finalizador para evitar exclusão acidental enquanto os volumes estão usando-as ativamente.

Passos

1. Excluir a política:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Se os volumes ainda estiverem usando a política, a exclusão entra em um `Deleting` estado. Verifique quais volumes são afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Remova a política de cada volume afetado. Escolha uma das seguintes opções:

   - **Opção A: desative explicitamente o crescimento automático** definindo a anotação como `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - **Opção B: remover a anotação completamente**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportamento de exclusão

Cenário	Comportamento
Nenhum volume utiliza a política	A política é excluída imediatamente.
Os volumes estão utilizando a política	A política entra em `Deleting` estado. Um finalizador bloqueia a conclusão até que todos os volumes sejam removidos.
Todos os volumes são removidos da política	Os finalizadores são removidos e a policy é excluída.

Monitorar o uso da política Autogrow

Verificar volumes usando uma política

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Descubra qual política um volume utiliza

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Monitorar eventos de políticas

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocolos suportados

Autogrow suporta os seguintes protocolos de armazenamento:

- NFS
- iSCSI
- FCP
- NVMe

> Para volumes SAN, se o `growthAmount` configurado for de 50 MiB ou menos, Trident aumenta automaticamente o valor de crescimento para 51 MB na operação de redimensionamento, desde que o tamanho resultante não exceda `maxSize`.

Limitações conhecidas

- **ONTAP Volumes de bloco bruto NVMe:** Volumes criados com versões do ONTAP anteriores à 9.16.1 não suportam autogrow.
- **Volumes existentes (implantações brownfield):** O Autogrow pode não funcionar para volumes existentes, mesmo que uma Política de Autogrow válida seja aplicada. Isso ocorre devido a uma migração em andamento das publicações de volume. Para confirmar se a migração foi concluída, verifique os logs do controlador Trident para mensagens `"Migration completed"`.

Perguntas frequentes

Quando o Trident avalia o limiar?

Trident monitora continuamente o uso do volume. Quando a capacidade utilizada ultrapassa o `usedThreshold`, Trident cria uma solicitação interna de redimensionamento e expande o volume pela configuração de `growthAmount`.

Por exemplo, esta política aciona a expansão em 80% da capacidade e aumenta o volume em 10% cada vez, até um máximo de 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Posso aplicar uma política depois que os volumes já foram provisionados?

Sim. Você pode criar uma Política de Crescimento Automático a qualquer momento e aplicá-la a PVCs existentes adicionando ou atualizando a `trident.netapp.io/autogrowPolicy` anotação. Você não precisa recriar o PVC ou o StorageClass.

Para aplicar uma política a um PVC existente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Para aplicar uma política a um StorageClass existente:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

O que acontece se eu definir uma Autogrow Policy tanto no StorageClass quanto no PVC?

A anotação do PVC sempre tem precedência. Se um PVC tiver a `trident.netapp.io/autogrowPolicy` anotação, Trident usa esse valor independentemente do que o StorageClass especifica. Consulte "Precedência de política" para obter detalhes.

Por exemplo, dado este StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

E este PVC que se sobrepõe à política de StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident usa `logs-policy` para `database-pvc`, não `standard-agp`.

Como faço para desativar o autogrow para um volume específico?

Defina a anotação PVC como `"none"`. Isso substitui qualquer política de nível de StorageClass para esse volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Você pode verificar se o autogrow está desativado:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
none
```

O que acontece quando um volume atinge maxSize?

Trident interrompe a expansão do volume. Nenhuma outra solicitação de redimensionamento é criada para esse volume, mesmo que o uso continue a aumentar além do `usedThreshold`.

Por exemplo, com essa política, Trident para de crescer o volume assim que ele atinge 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Para permitir crescimento ilimitado, omita `maxSize` ou defina como `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Posso alterar uma política sem reiniciar os volumes?

Sim. Quando você atualiza uma política, todos os volumes que usam essa política adotam os novos parâmetros na próxima avaliação de crescimento. Não é necessário reiniciar nenhum volume.

Para atualizar uma política existente:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifique os campos conforme necessário:

```
spec:
  usedThreshold: "75%"    # Changed from 80%
  growthAmount: "20%"     # Changed from 10%
  maxSize: "1Ti"          # Changed from 500Gi
```

Salve e saia. Verifique a política atualizada:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Resultado esperado

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

Por que minha policy está em estado de falha?

Um `Failed` estado indica que a especificação da política contém erros de validação. Execute o seguinte comando para visualizar os detalhes do erro:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

As causas comuns incluem um `usedThreshold` inválido (deve estar entre 1–99%), um `growthAmount` que excede `maxSize`, ou um formato de quantidade do Kubernetes inválido. Corrija a especificação e reaplique:

```
kubectl apply -f autogrow-policy.yaml
```

Por que não consigo excluir uma política?

As políticas utilizam proteção de finalização. Se os volumes ainda estiverem utilizando a política, a exclusão entra em um estado `Deleting` e aguarda até que todos os volumes sejam removidos da política.

Identifique os volumes afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Em seguida, remova a anotação de cada PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Depois que todos os volumes são removidos, o finalizador é liberado e a política é excluída.

O recurso de crescimento automático funciona com todos os backends do ONTAP?

Autogrow é compatível com os protocolos NFS, iSCSI, FCP e NVMe. No entanto, volumes de bloco bruto NVMe exigem ONTAP 9.16.1 ou posterior.

Os volumes existentes em implantações brownfield podem exigir que a migração da publicação de volume seja concluída antes que o autogrow entre em vigor. Verifique o status da migração consultando os logs do controlador Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Os seguintes exemplos de StorageClass mostram o autogrow configurado para backends NAS e SAN:

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Qual é o valor mínimo de crescimento para volumes SAN?

Para volumes SAN, o crescimento mínimo efetivo é de 51 MB. Se você configurar um `growthAmount` de 50 MiB ou menos, Trident aumenta automaticamente o crescimento para 51 MB na operação de redimensionamento.

Por exemplo, esta política define um `growthAmount` de `"40Mi"`, mas Trident aplica um crescimento de 51 MB para qualquer volume SAN que a utilize:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Para evitar esse ajuste automático, defina `growthAmount` para um valor superior a 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importar volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import` ou criando uma Persistent Volume Claim (PVC) com anotações de importação do Trident.

Visão geral e considerações

Você pode importar um volume para 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 aplicativo durante a recuperação de desastres

Considerações

Antes de importar um volume, revise as seguintes considerações.

- Trident pode importar apenas volumes ONTAP do tipo RW (leitura e gravação). 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 Trident.
- Sugerimos importar volumes sem conexões ativas. Para importar um volume em uso ativo, clone o volume e depois execute a importação.

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

- Embora `StorageClass` deva ser especificado em um PVC, Trident não utiliza esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação do volume para selecionar entre os pools disponíveis com base nas características de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo que o volume exista em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- O tamanho do volume existente é determinado e definido no PVC. Após o volume ser importado pelo driver de armazenamento, o PV é criado com um ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com sucesso o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Storage Class.
  - Se a política de recuperação da Storage Class for `delete`, o volume de armazenamento será excluído quando o PV for excluído.
- Por padrão, Trident gerencia o PVC e renomeia o FlexVol volume e o LUN no backend. Você pode usar a `--no-manage` flag para importar um volume não gerenciado e a `--no-rename` flag para manter o nome do volume.
  - `--no-manage`* - Se você usar a `--no-manage` flag, Trident não executa nenhuma operação adicional no PVC ou PV para o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído e outras operações, como clone de volume e expansão de volume, também são ignoradas.

- `--no-rename`* - Se você usar a `--no-rename`flag, Trident mantém o nome do volume existente ao importar volumes e gerencia o ciclo de vida dos volumes. Essa opção é compatível apenas para os drivers `ontap-nas`, `ontap-san` (incluindo sistemas ASA r2) e `ontap-san-economy`.

Essas opções são úteis se você deseja usar o Kubernetes para cargas de trabalho em contêineres, mas de outra forma deseja 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. Essa anotação não deve ser modificada ou removida.

Importar um volume

Você pode importar um volume usando `tridentctl import` ou criando um PVC com anotações de importação do Trident.

Se você utiliza anotações PVC, não precisa baixar ou usar `tridentctl` para importar o volume.

Usando tridentctl

Passos

1. Crie um arquivo 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.

   A seguir está um exemplo de especificação mínima:

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

   Inclua apenas os parâmetros necessários. Parâmetros adicionais, como o nome do PV ou o tamanho do volume, podem 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). O `-f` argumento é obrigatório para especificar o caminho para o arquivo PVC.

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

Usando anotações em PVC

Passos

1. Crie um arquivo YAML de PVC (por exemplo, `pvc.yaml`) com as anotações de importação do Trident necessárias. O arquivo de PVC deve incluir:

   - `name` e `namespace` em metadados
   - `accessModes`, `resources.requests.storage`, e `storageClassName` em especificação
   - Anotações:
     - `trident.netapp.io/importOriginalName`: Nome do volume no backend
     - `trident.netapp.io/importBackendUUID`: UUID do backend onde o volume existe
     - `trident.netapp.io/notManaged` (*Opcional*): Defina como `"true"` para volumes não gerenciados. O padrão é `"false"`.

       A seguir está um exemplo de especificação para importar um volume gerenciado:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Aplique o arquivo YAML do PVC ao seu cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automaticamente o volume e o vinculará ao PVC.

Exemplos

Revise os seguintes exemplos de importação de volume para drivers compatíveis.

ONTAP NAS e ONTAP NAS FlexGroup

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

- Trident não oferece suporte à importação de volumes usando o `ontap-nas-economy` driver.
- Os `ontap-nas` e `ontap-nas-flexgroup` drivers não permitem nomes de volume duplicados.

Cada volume criado com o `ontap-nas` driver é um volume FlexVol no cluster ONTAP. Importar volumes FlexVol com o `ontap-nas` 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, volumes FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.

Exemplos de ONTAP NAS usando tridentctl

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando `tridentctl`.

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

Exemplos do ONTAP NAS usando anotações PVC

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando anotações PVC.

Volume gerenciado

O exemplo a seguir importa um volume de 1GiB `ontap-nas` nomeado `ontap_volume1` do backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume não gerenciado

O exemplo a seguir importa 1Gi `ontap-nas` volume nomeado `ontap-volume2` do backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident suporta a importação de volumes usando os `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy`drivers.

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

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados:

Volume gerenciado

Para volumes gerenciados, Trident renomeia o FlexVol volume para o `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 erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desmapear a 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
```

Exemplos de SAN-economy do ONTAP

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados para o `ontap-san-economy` backend.

Volume gerenciado

Ao importar um volume gerenciado, Trident assume a propriedade do FlexVol e o renomeia. Você deve levar em consideração essa renomeação ao importar vários LUNs do mesmo FlexVol.

O exemplo a seguir importa `lun1` de FlexVol `toimport` como um volume gerenciado chamado `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Após a importação `lun1`, Trident renomeia o FlexVol (por exemplo, para `trident_lun_pool_xyz`). Para importar LUNs adicionais do mesmo FlexVol, use o novo nome do FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

O `ontap-san-economy` backend importa um LUN por vez. Você pode automatizar várias importações usando um script.

Volume não gerenciado

Ao importar um volume não gerenciado, Trident não assume a propriedade do FlexVol. No entanto, o FlexVol e o LUN devem seguir as convenções de nomenclatura do Trident.

Formato de nomenclatura do FlexVol

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` é o valor de `storagePrefix` na sua configuração de backend. O padrão é `trident`.
- `RANDOMSTRING` é qualquer sequência de caracteres que você escolher.

Requisito de nomenclatura LUN

O LUN deve ser nomeado `lun0`.

Exemplo

Se o seu `storagePrefix` é `xyz`, o caminho completo para o LUN é:

```
trident_lun_pool_xyz_randomstring/lun0
```

Elemento

Trident oferece suporte ao software NetApp Element e à importação de volumes NetApp HCI usando o driver `solidfire-san`.

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

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

Azure NetApp Files

Trident oferece suporte à importação de volumes usando o `azure-netapp-files` driver.

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

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

Trident oferece suporte à importação de volumes usando o `google-cloud-netapp-volumes` driver.

O exemplo a seguir importa um 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 Trident, você pode atribuir nomes e rótulos significativos aos volumes que cria. 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 volumes personalizados 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:

- Operações de criação, importação e clonagem de volume.
- No caso do `ontap-nas-economy`driver, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
- No caso do `ontap-san-economy`driver, apenas o nome do LUN está em conformidade com o modelo de nome.

Limitações

- Os nomes de volumes personalizados são compatíveis apenas com drivers ONTAP locais.
- Rótulos personalizados são suportados apenas para os `ontap-san`, `ontap-nas` e `ontap-nas-flexgroup` drivers.
- Nomes de volume personalizados não se aplicam a volumes existentes.

Principais comportamentos dos nomes de volume personalizáveis

- Se ocorrer uma falha devido à 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.
- 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 template 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 pool

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

Exemplos de 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 de 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. Trident não suporta o uso de um operador de slice com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, Trident adicionará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume econômico do NAS exceder 64 caracteres, 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.

Compartilhe um volume NFS entre namespaces

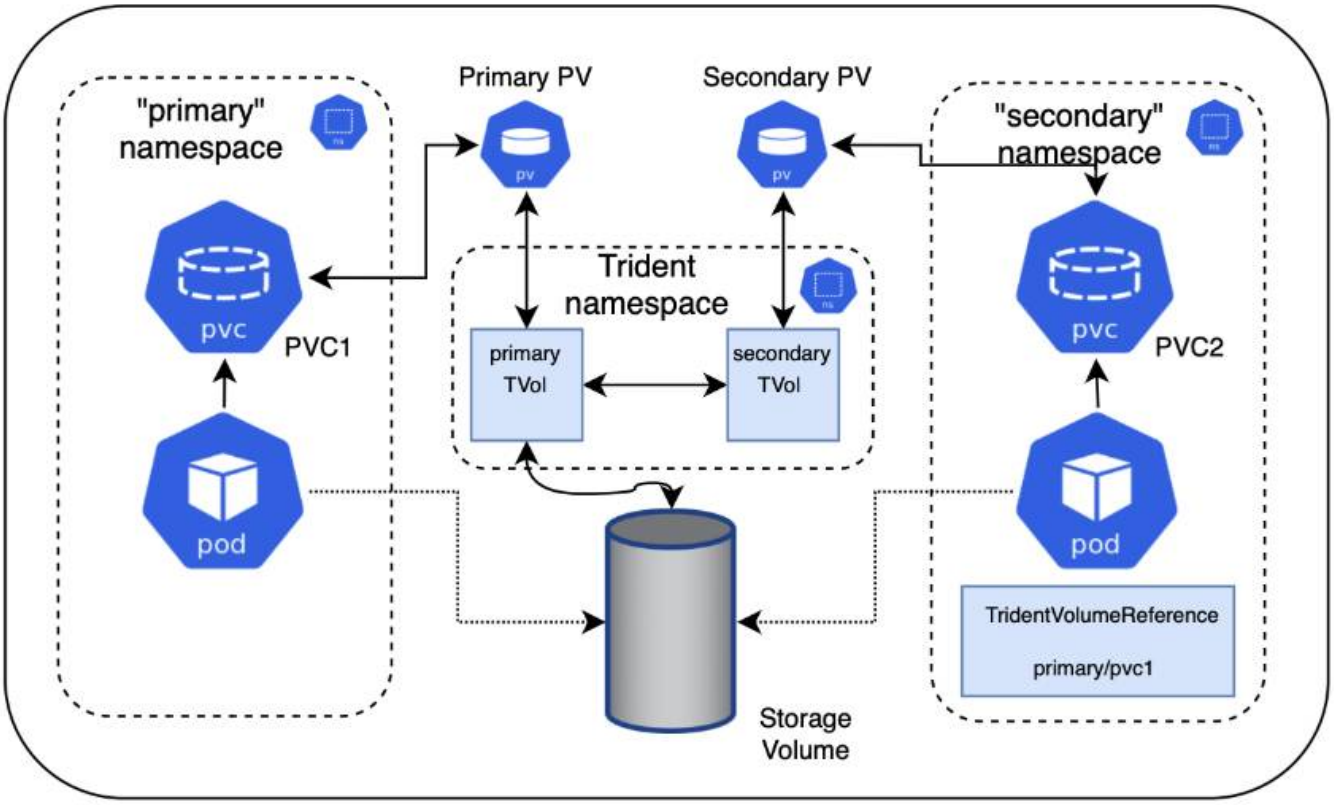
Usando Trident, você pode criar um volume em um namespace primário e compartilhá-lo em um ou mais namespaces secundários.

Características

O CR TridentVolumeReference permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) entre 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 Conceda permissão para criar um CR no namespace de destino

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

3 Criar TridentVolumeReference no namespace de destino

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência 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 designada em cada etapa.

Passos

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

```
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
```

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

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

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

Resultados

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

Excluir um volume compartilhado

Você pode excluir um volume compartilhado entre vários namespaces. 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, Trident exclui o volume.

Use `tridentctl get` para consultar volumes subordinados

Utilizando o [`tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para mais informações, consulte `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

- 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 do volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` ou `shareFromNamespace` anotações ou excluindo o `TridentVolumeReference` CR. Para revogar o acesso, é necessário excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para obter mais informações

Para saber mais sobre acesso a volumes entre namespaces:

- Veja a demonstração em "NetAppTV".

Clonar volumes entre namespaces

Usando Trident, você pode criar novos volumes usando volumes existentes ou volumesnapshots 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 os `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.

1. **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.

2 **Conceda permissão para criar um CR no namespace de destino**

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

3 **Criar TridentVolumeReference no namespace de destino**

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência ao PVC de origem.

4 **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 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 designada em cada etapa.

Passos

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

```
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
```

Trident cria o PV e seu volume de armazenamento 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 a `cloneToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja configurado 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 um 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:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando as `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 provisionados usando drivers ontap-nas-economy, clones somente leitura não são suportados.

Replicar volumes usando SnapMirror

Trident oferece suporte a 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 realizar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova relações de espelhamento entre volumes
- Interrompa 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 de replicação

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

Clusters do ONTAP

- **Trident**: A versão 22.10 ou posterior do Trident deve existir em ambos os clusters Kubernetes de origem e destino que utilizam ONTAP como backend.
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote Data Protection devem estar habilitadas tanto nos clusters ONTAP de origem quanto de destino. Consulte "SnapMirror visão geral do licenciamento 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 com o ONTAP One" para mais informações.

  > Apenas a proteção assíncrona SnapMirror é suportada.

Peering

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar emparelhados. Consulte "Visão geral do peering de cluster e SVM" 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 Trident no cluster de destino.

Drivers com suporte

NetApp Trident suporta replicação de volumes com a tecnologia NetApp SnapMirror usando 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 volume usando SnapMirror não é suportada para 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 relação de espelhamento entre os volumes primário e secundário.

Passos

1. Execute as seguintes etapas no cluster Kubernetes primário:
   a. Crie um objeto 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:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Crie um PVC com StorageClass previamente criado.

      Exemplo

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

   c. Crie um MirrorRelationship CR com informações locais.

Exemplo

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

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

d. Obtenha o TridentMirrorRelationship CR 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 as seguintes etapas no cluster Kubernetes secundário:

   a. Crie um 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 CR 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"
```

Trident criará um SnapMirror relacionamento com o nome da política de relacionamento configurada (ou padrão para ONTAP) e o inicializará.

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

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
```

Trident verificará o CRD TridentMirrorRelationship e não conseguirá criar o volume se o relacionamento não existir. Se o relacionamento existir, Trident garantirá que o novo volume FlexVol seja colocado em uma SVM que esteja emparelhada com a SVM remota definida no MirrorRelationship.

Estados de replicação de volume

Um Trident Mirror Relationship (TMR) é um CRD que representa uma das extremidades de uma relação de replicação entre PVCs. 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.
- **Promovido**: o PVC local é ReadWrite e montável, sem 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 estava previamente em uma relação com a origem.

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 um failover (migração) planejado, execute as seguintes etapas 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 SnapshotInfo CR para obter detalhes internos.

   Exemplo

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

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

Passos

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

Operações adicionais

Trident suporta as seguintes operações nos volumes primário e secundário:

Replicar o PVC primário em um novo PVC secundário

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

Passos

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

Redimensione um PVC espelhado, primário ou secundário

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

Remover replicação de um PVC

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

- Exclua o MirrorRelationship no PVC secundário. Isso interrompe a relação de replicação.
- Ou, atualize o campo spec.state para *promoted*.

Excluir um PVC (que foi previamente espelhado)

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 um relacionamento espelhado existente e o TMR será removido e o Trident promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados SnapMirror do volume local no ONTAP. Se esse volume for usado em um relacionamento espelhado no futuro, será necessário usar um novo TMR com um estado de replicação de volume *estabelecido* ao criar o novo relacionamento espelhado.

Atualize os relacionamentos de espelhamento quando 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 as relações. Ao promover um volume de destino para um volume ReadWrite regular, você pode usar *promotedSnapshotHandle* para especificar um snapshot específico para restaurar o volume atual.

Atualizar relações de espelhamento quando ONTAP estiver offline

Você pode usar um CRD para realizar uma atualização 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 TridentActionMirrorUpdate CRD. Pode assumir um valor de *Succeeded*, *In Progress* ou *Failed*.

Usar a topologia CSI

Trident pode criar e anexar volumes seletivamente aos nós presentes em um cluster Kubernetes fazendo uso do "Recurso de topologia CSI".

Visão geral

Utilizando o recurso de CSI Topology, o acesso a 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, Trident utiliza CSI Topology.

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

Kubernetes oferece dois modos exclusivos de vinculação de volumes:

- Com `VolumeBindingMode` definido como `Immediate`, Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume 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 depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um Volume Persistente para um PVC são adiadas até que um pod que utiliza 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`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 a consciência de topologia (`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Esses rótulos **devem estar presentes nos nós do cluster** antes que o Trident seja instalado para que o Trident 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"}]
```

Etapa 1: crie um backend com reconhecimento de topologia

Os backends de storage Trident podem ser projetados para provisionar volumes seletivamente com base em zonas de disponibilidade. Cada backend pode conter um bloco `supportedTopologies` opcional que representa uma lista de zonas e regiões suportadas. Para StorageClasses que fazem uso de tal backend, um volume só será criado se solicitado por uma aplicação agendada 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 usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um StorageClass. Para StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria um volume no backend.

Você pode definir `supportedTopologies` por pool de storage 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, os `region` e `zone` rótulos representam a localização do pool de storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` indicam de onde os pools de storage podem ser consumidos.

Etapa 2: Defina StorageClasses que são cientes da topologia

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

Veja o exemplo a seguir:

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

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

Etapa 3: criar e usar um PVC

Com o StorageClass criado e mapeado 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 Trident crie um volume e o vincule ao PVC, use o PVC em um pod. Veja o exemplo a seguir:

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

Este podSpec instrui o Kubernetes a agendar o pod em nós que estão presentes na `us-east1` região e escolher entre qualquer nó que esteja presente nas `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
```

Atualize 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"
- "nodeSelector"
- "Afinidade e antiafinidade"
- "Taints e Tolerations"

Trabalhar com Snapshots

Os snapshots de volumes persistentes (PVs) do Kubernetes permitem cópias de volumes em ponto no tempo. Você pode criar um snapshot de um volume criado usando 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 pelos `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` e `google-cloud-netapp-volumes` drivers.

Antes de começar

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

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, consulte Implante um controlador de Snapshot de volume.

> Não crie um controlador de snapshot se estiver criando snapshots de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

Criar um snapshot de volume

Passos

1. Crie um `VolumeSnapshotClass`. Para obter mais informações, consulte "VolumeSnapshotClass".
   - O `driver` aponta para o driver Trident CSI.
   - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o snapshot físico subjacente no cluster de armazenamento é mantido mesmo quando o objeto `VolumeSnapshot` é excluído.

     Exemplo

     ```
     cat snap-sc.yaml
     ```

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

2. Crie um snapshot de um PVC existente.

   Exemplos

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

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

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

- Você pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o. O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve a este snapshot. O `Ready To Use` parâmetro indica que o snapshot pode ser usado para criar um novo PVC.

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

Crie um PVC a partir de um instantâneo de volume

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

> O PVC será criado no mesmo backend que o volume de origem. Consulte "KB: A criação de um PVC a partir de um Trident PVC Snapshot 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 Snapshot de volume

Trident oferece suporte à "Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador do cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Trident.

Antes de começar

Trident deve ter criado ou importado o volume pai do Snapshot.

Passos

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

     > O `<volumeSnapshotContentName>` não pode sempre corresponder ao nome do snapshot do backend devido a restrições de nomenclatura do CR.

     Exemplo

     O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência ao 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 referencia o `VolumeSnapshotContent` objeto. Isso solicita acesso para usar o `VolumeSnapshot` em um determinado namespace.

   Exemplo

   O exemplo a seguir cria um `VolumeSnapshot` CR chamado `import-snap` que faz referência ao `VolumeSnapshotContent` chamado `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 a `ListSnapshots` chamada. Trident cria o `TridentSnapshot`.
   - O snapshotter externo define o `VolumeSnapshotContent` para `readyToUse` e o `VolumeSnapshot` para `true`.
   - Trident está de volta `readyToUse=true`.
4. **Qualquer usuário:** Crie um `PersistentVolumeClaim` para referenciar o novo `VolumeSnapshot`, onde o `spec.dataSource` (ou `spec.dataSourceRef`) nome é o nome `VolumeSnapshot`.

   Exemplo

O exemplo a seguir cria um PVC referenciando o `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 está oculto por padrão para facilitar a máxima compatibilidade de volumes provisionados usando os `ontap-nas` e `ontap-nas-economy` drivers. Habilite o `.snapshot` diretório para recuperar dados diretamente dos snapshots.

Use o volume snapshot restore ONTAP CLI 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 do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Trident suporta a restauração de snapshot nos `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` e `solidfire-san` drivers.

Antes de começar

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

- Verifique se o status do PVC está bound.

```
kubectl get pvc
```

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

```
kubectl get vs
```

Passos

1. Crie o CR TASR. Este exemplo cria um CR para PVC `pvc1` e volume snapshot `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 CR para restaurar a partir do snapshot. Este exemplo restaura a partir do snapshot `pvc1`.

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

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

Resultados

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, Trident não tentará novamente a operação automaticamente em caso de falha. Você precisará executar a operação novamente.
- Usuários do Kubernetes sem acesso de administrador podem precisar receber permissão do administrador para criar um TASR CR em seu namespace de aplicação.

Excluir um PV com snapshots associados

Ao excluir um Persistent Volume com snapshots associados, o volume Trident correspondente é atualizado para o estado "Deleting state". Remova os snapshots do volume para excluir o volume Trident.

Implante um controlador de Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller 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 snapshot.

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

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

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

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Trabalhar com Snapshots de grupo de volume

Snapshots de grupo de volumes do Kubernetes de Persistent Volumes (PVs) NetApp Trident fornecem a capacidade de criar snapshots de múltiplos volumes (um grupo de snapshots de volumes). Esse snapshot de grupo de volumes representa cópias de múltiplos volumes feitas no mesmo ponto no tempo.

VolumeGroupSnapshot é um recurso beta no Kubernetes com APIs beta. Kubernetes 1.32 é a versão mínima necessária para VolumeGroupSnapshot.

Criar snapshots de grupo de volume

O snapshot de grupo de volume é compatível com os seguintes drivers de armazenamento:

- `ontap-san`driver - somente para os protocolos iSCSI e FC, não para o protocolo NVMe/TCP.
- `ontap-san-economy` - somente para o protocolo iSCSI.
- `ontap-nas`

> O snapshot de grupo de volume não é compatível com sistemas de armazenamento NetApp ASA r2 ou AFX.

Antes de começar

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

  Se a sua distribuição Kubernetes não incluir o controlador de snapshot externo e os CRDs, consulte Implante um controlador de Snapshot de volume.

  > Não crie um controlador de snapshot se estiver criando snapshots de grupo de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

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

Passos

- Crie um 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 o rótulo `consistentGroupSnapshot: groupA`. Defina a chave e o valor do rótulo de acordo com 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 Snapshot de grupo de volume:

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

Recuperar dados de volume usando um snapshot de grupo

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

Use o volume snapshot restore ONTAP CLI 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 do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Para obter mais informações, consulte "Restauração de volume in-place a partir de uma Snapshot".

Excluir um PV com snapshots de grupo associados

Ao excluir um Snapshot de grupo de volume:

- Você pode excluir VolumeGroupSnapshots como um todo, não snapshots individuais do grupo.
- Se PersistentVolumes forem excluídos enquanto existir um snapshot para esse PersistentVolume, Trident moverá esse volume para um estado de "exclusão", pois o snapshot precisa 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 Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller 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 snapshot.

```
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 atualize `namespace` para o seu namespace.

Links relacionados

- "VolumeGroupSnapshotClass"
- "Instantâneos de volume"

Informações sobre direitos autorais

Informações sobre marcas comerciais