볼륨 제공 및 관리

Trident

NetApp
March 05, 2026

목차

볼륨 제공 및 관리

볼륨 제공

구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim(PVC)을 만듭니다. 그런 다음 PV를 포드에 장착할 수 있습니다.

개요

에이 "지속적 볼륨 클레임" (PVC)는 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

PVC는 특정 크기 또는 액세스 모드의 저장소를 요청하도록 구성될 수 있습니다. 연관된 StorageClass를 사용하면 클러스터 관리자는 PersistentVolume 크기 및 액세스 모드(성능이나 서비스 수준 등) 이상을 제어할 수 있습니다.

PVC를 만든 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 만듭니다

단계

1. PVC을 생성합니다.

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

2. PVC 상태를 확인하세요.

```
kubectl get pvc
```

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

1. 볼륨을 포드에 마운트합니다.

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

> 다음을 사용하여 진행 상황을 모니터링할 수 있습니다. `kubectl get pod --watch` .

2. 볼륨이 마운트되었는지 확인하세요. `/my/mount/path` .

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

3. 이제 Pod를 삭제할 수 있습니다. Pod 애플리케이션은 더 이상 존재하지 않지만 볼륨은 그대로 유지됩니다.

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

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

다음 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWO 접근이 가능한 **PVC**

이 예에서는 StorageClass와 연관된 RWO 액세스가 있는 기본 PVC를 보여줍니다. `basic-csi` .

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

NVMe/TCP를 사용한 **PVC**

이 예에서는 RWO 액세스가 있는 NVMe/TCP용 기본 PVC를 보여줍니다. 이 PVC는 StorageClass와 연결되어 있습니다. `protection-gold` .

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

Pod 매니페스트 샘플

이러한 예는 PVC를 포드에 부착하는 기본 구성을 보여줍니다.

기본 구성

```
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
```

기본 **NVMe/TCP** 구성

```
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
```

참조하다"Kubernetes 및 Trident 객체" 저장소 클래스가 어떻게 상호 작용하는지에 대한 자세한 내용은 다음과 같습니다. `PersistentVolumeClaim` Trident 가 볼륨을 프로비저닝하는 방식을 제어하기 위한 매개변수입니다.

볼륨 확장

Trident Kubernetes 사용자에게 볼륨을 생성한 후에 볼륨을 확장할 수 있는 기능을 제공합니다. iSCSI, NFS, SMB, NVMe/TCP 및 FC 볼륨을 확장하는 데 필요한 구성에 대한 정보를 찾아보세요.

iSCSI 볼륨 확장

CSI 프로비저너를 사용하여 iSCSI 영구 볼륨(PV)을 확장할 수 있습니다.

> iSCSI 볼륨 확장은 다음에서 지원됩니다. `ontap-san` , `ontap-san-economy` , `solidfire-san` 드라이버가 필요하며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

StorageClass 정의를 편집하여 다음을 설정합니다. `allowVolumeExpansion` 필드로 `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
```

이미 존재하는 StorageClass의 경우 다음을 포함하도록 편집합니다. `allowVolumeExpansion` 매개변수.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다**.**

PVC 정의를 편집하고 업데이트합니다. `spec.resources.requests.storage` 새로 원하는 크기를 반영해야 하며, 원래 크기보다 커야 합니다.

```
cat pvc-ontapsan.yaml
```

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

Trident 영구 볼륨(PV)을 생성하고 이를 영구 볼륨 클레임(PVC)과 연결합니다.

```
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
```

3단계: **PVC**를 부착하는 포드 정의

PV를 포드에 부착하여 크기를 조절할 수 있습니다. iSCSI PV 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결되어 있으면 Trident 스토리지 백엔드의 볼륨을 확장하고, 장치를 다시 스캔하고, 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident 스토리지 백엔드의 볼륨을 확장합니다. PVC가 포드에 바인딩되면 Trident 장치를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 다음을 사용하는 포드가 생성됩니다. `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
```

4단계: **PV** 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 업데이트하십시오. `spec.resources.requests.storage` 2Gi로.

```
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
 # ...
```

5단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 작동하는지 확인할 수 있습니다.

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

FC 볼륨 확장

CSI 프로비저너를 사용하여 FC 영구 볼륨(PV)을 확장할 수 있습니다.

FC 볼륨 확장은 다음에 의해 지원됩니다. `ontap-san` 드라이버이며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

StorageClass 정의를 편집하여 다음을 설정합니다. `allowVolumeExpansion` 필드로 `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
```

이미 존재하는 StorageClass의 경우 다음을 포함하도록 편집합니다. `allowVolumeExpansion` 매개변수.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다.

PVC 정의를 편집하고 업데이트합니다. `spec.resources.requests.storage` 새로 원하는 크기를 반영해야 하며, 원래 크기보다 커야 합니다.

```
cat pvc-ontapsan.yaml
```

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

Trident 영구 볼륨(PV)을 생성하고 이를 영구 볼륨 클레임(PVC)과 연결합니다.

```
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
```

3단계**: PVC**를 부착하는 포드 정의

PV를 포드에 부착하여 크기를 조절할 수 있습니다. FC PV 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결되어 있으면 Trident 스토리지 백엔드의 볼륨을 확장하고, 장치를 다시 스캔하고, 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident 스토리지 백엔드의 볼륨을 확장합니다. PVC가 포드에 바인딩되면 Trident 장치를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 다음을 사용하는 포드가 생성됩니다. `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
```

4단계: PV 확장

1Gi에서 2Gi로 생성된 PV의 크기를 조정하려면 PVC 정의를 편집하고 업데이트하십시오. `spec.resources.requests.storage` 2Gi로.

```
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
 # ...
```

5단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 확장이 올바르게 작동하는지 확인할 수 있습니다.

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

NFS 볼륨 확장

Trident NFS PV에 대한 볼륨 확장을 지원합니다. `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `gcp-cvs` , 그리고 `azure-netapp-files` 백엔드.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

NFS PV 크기를 조정하려면 관리자는 먼저 볼륨 확장을 허용하도록 스토리지 클래스를 구성해야 합니다. `allowVolumeExpansion` 필드로 `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
```

이 옵션 없이 이미 스토리지 클래스를 생성한 경우 다음을 사용하여 기존 스토리지 클래스를 간단히 편집할 수 있습니다. `kubectl edit storageclass` 볼륨 확장을 허용합니다.

2단계: 생성한 **StorageClass**로 **PVC**를 생성합니다.

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

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

Trident 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
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
```

3단계: **PV** 확장

새로 생성된 20 MiB PV를 1 GiB로 크기를 조정하려면 PVC를 편집하고 설정하세요. `spec.resources.requests.storage` 1GiB까지:

```
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
# ...
```

4단계: 확장 검증

PVC, PV 및 Trident 볼륨의 크기를 확인하여 크기 조정이 올바르게 수행되었는지 확인할 수 있습니다.

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

수입량

기존 스토리지 볼륨을 Kubernetes PV로 가져오려면 `tridentctl import`를 사용하거나 Trident 가져오기 어노테이션을 사용하여 영구 볼륨 클레임(PVC)을 생성할 수 있습니다.

개요 및 고려 사항

볼륨을 Trident 로 가져와서 다음을 수행할 수 있습니다.

- 애플리케이션을 컨테이너화하고 기존 데이터 세트를 재사용합니다.
- 임시 애플리케이션에 데이터 세트 복제본 사용
- 실패한 Kubernetes 클러스터 재구축
- 재해 복구 중 애플리케이션 데이터 마이그레이션

고려 사항

볼륨을 가져오기 전에 다음 고려 사항을 검토하세요.

- Trident RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror

대상 볼륨입니다. 볼륨을 Trident 로 가져오기 전에 미러 관계를 해제해야 합니다.

- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 현재 사용되는 볼륨을 가져오려면 볼륨을 복제한 다음 가져오기를 수행합니다.

이는 블록 볼륨의 경우 특히 중요합니다. Kubernetes는 이전 연결을 인식하지 못하고 활성 볼륨을 Pod에 쉽게 연결할 수 있기 때문입니다. 이로 인해 데이터가 손상될 수 있습니다.

- 그렇지만 `StorageClass` PVC에서는 이 매개변수를 지정해야 하지만, Trident 가져오기 중에 이 매개변수를 사용하지 않습니다. 스토리지 클래스는 볼륨을 생성하는 동안 스토리지 특성에 따라 사용 가능한 풀을 선택하는 데 사용됩니다. 볼륨이 이미 존재하므로 가져오는 동안 풀을 선택할 필요가 없습니다. 따라서 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드나 풀에 볼륨이 존재하더라도 가져오기가 실패하지 않습니다.
- 기존 볼륨 크기가 결정되어 PVC에 설정됩니다. 볼륨이 스토리지 드라이버에 의해 가져온 후, PV는 PVC에 대한 ClaimRef와 함께 생성됩니다.
  - 회수 정책은 처음에 다음과 같이 설정됩니다. `retain` PV에서. Kubernetes가 PVC와 PV를 성공적으로 바인딩한 후, 회수 정책이 스토리지 클래스의 회수 정책과 일치하도록 업데이트됩니다.
  - 스토리지 클래스의 회수 정책이 있는 경우 `delete` PV가 삭제되면 저장 볼륨도 삭제됩니다.
- 기본적으로 Trident PVC를 관리하고 백엔드에서 FlexVol volume 과 LUN의 이름을 변경합니다. 당신은 통과 할 수 있습니다 `--no-manage` 관리되지 않는 볼륨을 가져오기 위한 플래그입니다. 당신이 사용하는 경우 `--no-manage` Trident 객체의 수명 주기 동안 PVC 또는 PV에 대한 추가 작업을 수행하지 않습니다. PV가 삭제되어도 저장 볼륨은 삭제되지 않으며 볼륨 복제 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다.

이 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하지만 그렇지 않은 경우 Kubernetes 외부에서 스토리지 볼륨의 수명 주기를 관리하려는 경우에 유용합니다.

- PVC와 PV에 주석이 추가되어 볼륨이 가져왔는지, PVC와 PV가 관리되는지 여부를 나타내는 두 가지 목적을 갖습니다. 이 주석은 수정하거나 제거해서는 안 됩니다.

볼륨 가져오기

`tridentctl import`를 사용하거나 `Trident` 가져오기 주석이 포함된 `PVC`를 생성하여 볼륨을 가져올 수 있습니다.

PVC 주석을 사용하는 경우 볼륨을 가져오기 위해 `tridentctl`를 다운로드하거나 사용할 필요가 없습니다.

tridentctl 사용

단계

1. PVC를 생성하는 데 사용할 PVC 파일(예: `pvc.yaml`)을 생성합니다. PVC 파일에는 `name`, `namespace`, `accessModes` 및 `storageClassName`가 포함되어야 합니다. 선택적으로 PVC 정의에서 `unixPermissions`를 지정할 수 있습니다.

   다음은 최소 사양의 예입니다.

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

   필수 매개변수만 포함하십시오. PV 이름이나 볼륨 크기와 같은 추가 매개변수는 가져오기 명령이 실패할 수 있습니다.

2. 사용하다 `tridentctl import` 볼륨을 포함하는 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume, Cloud Volumes Service 경로)을 지정하는 명령입니다. 그만큼 `-f` PVC 파일의 경로를 지정하려면 인수가 필요합니다.

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

PVC 주석 사용

단계

1. 필요한 Trident 가져오기 주석이 포함된 PVC YAML 파일(예: `pvc.yaml`)을 생성합니다. PVC 파일에는 다음 내용이 포함되어야 합니다.

   - `name` 및 `namespace` 메타데이터
   - `accessModes`, `resources.requests.storage` 및 `storageClassName` 사양
   - 주석:
     - `trident.netapp.io/importOriginalName`: 백엔드의 볼륨 이름
     - `trident.netapp.io/importBackendUUID`: 볼륨이 존재하는 백엔드 UUID
     - `trident.netapp.io/notManaged` (선택 사항): 관리되지 않는 볼륨의 경우 `"true"`로 설정합니다. 기본값은 `"false"`입니다.

     다음은 관리형 볼륨을 가져오기 위한 예시 사양입니다.

```
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. PVC YAML 파일을 Kubernetes 클러스터에 적용합니다.

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

Trident는 볼륨을 자동으로 가져와 PVC에 바인딩합니다.

예시

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하세요.

ONTAP NAS 및 ONTAP NAS FlexGroup

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `ontap-nas` 그리고 `ontap-nas-flexgroup` 운전자.

- Trident 다음을 사용하여 볼륨 가져오기를 지원하지 않습니다. `ontap-nas-economy` 운전사.
- 그만큼 `ontap-nas` 그리고 `ontap-nas-flexgroup` 드라이버는 중복된 볼륨 이름을 허용하지 않습니다.

각 볼륨은 다음과 같이 생성됩니다. `ontap-nas` 드라이버는 ONTAP 클러스터의 FlexVol volume 입니다. FlexVol 볼륨을 가져오는 방법 `ontap-nas` 드라이버도 동일하게 작동합니다. ONTAP 클러스터에 이미 존재하는 FlexVol 볼륨은 다음과 같이 가져올 수 있습니다. `ontap-nas` PVC. 마찬가지로 FlexGroup 볼륨은 다음과 같이 가져올 수 있습니다. `ontap-nas-flexgroup` PVC.

tridentctl을 사용한 ONTAP NAS 예제

다음은 관리되는 볼륨과 관리되지 않는 볼륨 가져오기의 예를 보여줍니다.

관리되는 볼륨

다음 예제에서는 이름이 지정된 볼륨을 가져옵니다. `managed_volume` 백엔드에 이름이 지정됨 `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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

사용 시 `--no-manage` 인수에 따라 Trident 볼륨의 이름을 바꾸지 않습니다.

다음 예제에서는 다음을 가져옵니다. `unmanaged_volume` 에 `ontap_nas` 백엔드:

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

PVC 주석을 사용한 **ONTAP NAS** 예제

다음 예제는 PVC 주석을 사용하여 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리되는 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `81abcb27-ea63-49bb-b606-0a5315ac5f21`에서 `ontap_volume1`라는 이름의 `1GiB` `ontap-nas` 볼륨을 가져옵니다.

```
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>
```

관리되지 않는 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `34abcb27-ea63-49bb-b606-0a5315ac5f34`에서 이름이 `ontap-volume2`인 `1Gi` `ontap-nas` 볼륨을 가져옵니다.

```
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 산

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `ontap-san` (iSCSI, NVMe/TCP 및 FC) 및 `ontap-san-economy` 운전자.

Trident 단일 LUN을 포함하는 ONTAP SAN FlexVol 볼륨을 가져올 수 있습니다. 이는 다음과 일치합니다. `ontap-san` 각 PVC에 대한 FlexVol volume 과 FlexVol volume 내의 LUN을 생성하는 드라이버입니다. Trident FlexVol volume 가져와 PVC 정의와 연결합니다. Trident 수입이 가능합니다 `ontap-san-economy` 여러 LUN을 포함하는 볼륨.

ONTAP SAN 예시

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리되는 볼륨

관리되는 볼륨의 경우 Trident FlexVol volume 이름을 다음과 같이 변경합니다. `pvc-<uuid>` FlexVol volume 내의 LUN과 포맷 `lun0` .

다음 예제에서는 다음을 가져옵니다. `ontap-san-managed` FlexVol volume 이 존재합니다. `ontap_san_default` 백엔드:

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

관리되지 않는 볼륨

다음 예제에서는 다음을 가져옵니다. `unmanaged_example_volume` 에 `ontap_san` 백엔드:

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

다음 예와 같이 Kubernetes 노드 IQN과 IQN을 공유하는 igroup에 LUN이 매핑된 경우 오류가 발생합니다. `LUN already mapped to initiator(s) in this group` . 볼륨을 가져오려면 초기자를 제거하거나 LUN의 매핑을 해제해야 합니다.

```
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
```

요소

Trident 다음을 사용하여 NetApp Element 소프트웨어 및 NetApp HCI 볼륨 가져오기를 지원합니다. `solidfire-san` 운전사.

> Element 드라이버는 중복된 볼륨 이름을 지원합니다. 그러나 볼륨 이름이 중복되면 Trident 오류를 반환합니다. 해결 방법으로 볼륨을 복제하고 고유한 볼륨 이름을 제공한 다음 복제된 볼륨을 가져옵니다.

다음 예제에서는 다음을 가져옵니다. `element-managed` 백엔드의 볼륨 `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     |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+----------+
```

구글 클라우드 플랫폼

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `gcp-cvs` 운전사.

> Google Cloud Platform에서 NetApp Cloud Volumes Service 지원하는 볼륨을 가져오려면 볼륨 경로로 볼륨을 식별합니다. 볼륨 경로는 볼륨 내보내기 경로의 일부입니다. `:/` . 예를 들어, 내보내기 경로가 `10.0.0.1:/adroit-jolly-swift` , 볼륨 경로는 `adroit-jolly-swift` .

Google Cloud Platform 예시

다음 예제에서는 다음을 가져옵니다. `gcp-cvs` 백엔드의 볼륨 `gcpcvs_YEppr` 볼륨 경로와 함께 `adroit-jolly-swift` .

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

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

Azure NetApp Files

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `azure-netapp-files` 운전사.

> Azure NetApp Files 볼륨을 가져오려면 볼륨 경로로 볼륨을 식별합니다. 볼륨 경로는 볼륨 내보내기 경로의 일부입니다. `:/` . 예를 들어, 마운트 경로가 `10.0.0.2:/importvol1` , 볼륨 경로는 `importvol1` .

다음 예제에서는 다음을 가져옵니다. `azure-netapp-files` 백엔드의 볼륨 `azurenetappfiles_40517` 볼륨 경로로 `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 다음을 사용하여 볼륨 가져오기를 지원합니다. `google-cloud-netapp-volumes` 운전사.

다음 예제는 backend `backend-tbc-gcnv1`에서 volume `testvoleasiaeast1`을 가져옵니다.

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

다음 예제에서는 다음을 가져옵니다. `google-cloud-netapp-volumes` 동일한 영역에 두 개의 볼륨이 존재할 때의 볼륨:

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

볼륨 이름 및 레이블 사용자 정의

Trident 사용하면 생성한 볼륨에 의미 있는 이름과 레이블을 지정할 수 있습니다. 이를 통해

볼륨을 식별하고 해당 Kubernetes 리소스(PVC)에 쉽게 매핑할 수 있습니다. 사용자 정의 볼륨 이름과 사용자 정의 레이블을 생성하기 위해 백엔드 수준에서 템플릿을 정의할 수도 있습니다. 생성, 가져오기 또는 복제하는 모든 볼륨은 템플릿을 준수합니다.

시작하기 전에

사용자 정의 가능한 볼륨 이름 및 레이블 지원:

1. 볼륨 생성, 가져오기 및 복제 작업.
2. ontap-nas-economy 드라이버의 경우, Qtree 볼륨의 이름만 이름 템플릿을 따릅니다.
3. ontap-san-economy 드라이버의 경우 LUN 이름만 이름 템플릿을 따릅니다.

제한 사항

1. 사용자 정의 가능한 볼륨 이름은 ONTAP 온프레미스 드라이버와만 호환됩니다.
2. 사용자 정의 가능한 볼륨 이름은 기존 볼륨에는 적용되지 않습니다.

사용자 정의 가능한 볼륨 이름의 주요 동작

1. 이름 템플릿의 구문이 잘못되어 오류가 발생하면 백엔드 생성이 실패합니다. 그러나 템플릿 애플리케이션이 실패하면 볼륨 이름은 기존 명명 규칙에 따라 지정됩니다.
2. 백엔드 구성의 이름 템플릿을 사용하여 볼륨의 이름을 지정한 경우 스토리지 접두사는 적용되지 않습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블을 사용한 백엔드 구성 예

사용자 정의 이름 템플릿은 루트 및/또는 풀 수준에서 정의할 수 있습니다.

루트 수준 예제

```
{
  "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}}"
  }
}
```

풀 레벨 예시

```
{
  "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 }}"
      }
    }
  ]
}
```

이름 템플릿 예시

예시 **1**:

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

예시 **2**:

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

고려할 사항

1. 볼륨 가져오기의 경우, 기존 볼륨에 특정 형식의 레이블이 있는 경우에만 레이블이 업데이트됩니다. 예를 들어: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. 관리형 볼륨 가져오기의 경우 볼륨 이름은 백엔드 정의의 루트 수준에서 정의된 이름 템플릿을 따릅니다.
3. Trident 저장 접두사와 함께 슬라이스 연산자를 사용하는 것을 지원하지 않습니다.
4. 템플릿에서 고유한 볼륨 이름이 생성되지 않으면 Trident 몇 개의 무작위 문자를 추가하여 고유한 볼륨 이름을 생성합니다.
5. NAS Economy 볼륨의 사용자 지정 이름이 64자를 초과하는 경우 Trident 기존 명명 규칙에 따라 볼륨 이름을 지정합니다. 다른 모든 ONTAP 드라이버의 경우 볼륨 이름이 이름 제한을 초과하면 볼륨 생성 프로세스가 실패합니다.

네임스페이스 간에 **NFS** 볼륨 공유

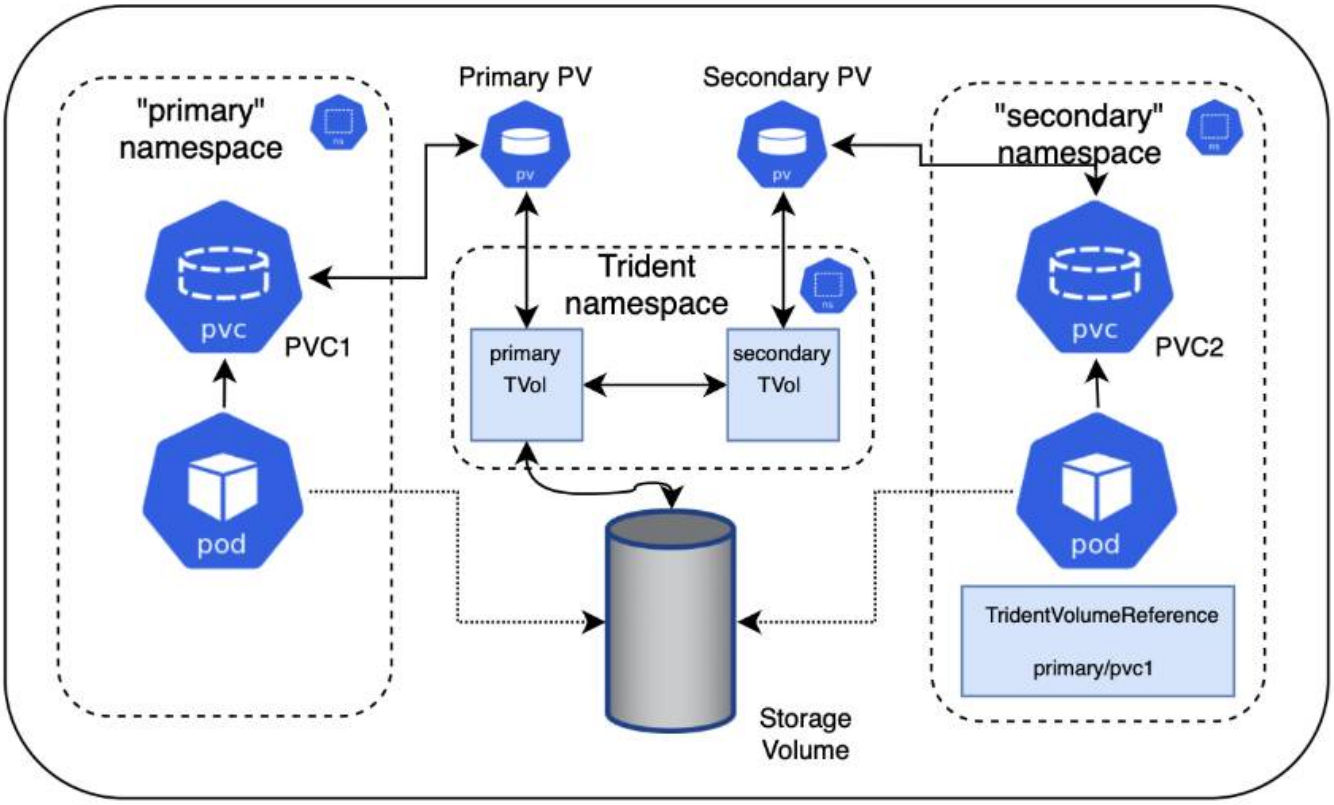
Trident 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

특징

TridentVolumeReference CR을 사용하면 하나 이상의 Kubernetes 네임스페이스에서 ReadWriteMany(RWX) NFS 볼륨을 안전하게 공유할 수 있습니다. 이 Kubernetes 기반 솔루션은 다음과 같은 이점을 제공합니다.

- 보안을 보장하기 위한 다양한 수준의 액세스 제어
- 모든 Trident NFS 볼륨 드라이버와 함께 작동합니다.
- tridentctl이나 기타 비네이티브 Kubernetes 기능에 대한 의존성 없음

이 다이어그램은 두 개의 Kubernetes 네임스페이스에서 NFS 볼륨을 공유하는 것을 보여줍니다.



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1. 볼륨을 공유하도록 소스 **PVC**를 구성합니다**.**

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.

2. 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여합니다**.**

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3. 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다**.**

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4. 대상 네임스페이스에 하위 **PVC**를 만듭니다**.**

대상 네임스페이스의 소유자는 소스 PVC의 데이터 소스를 사용하기 위해 하위 PVC를 생성합니다.

소스 및 대상 네임스페이스 구성

보안을 보장하려면 네임스페이스 간 공유에 소스 네임스페이스 소유자, 클러스터 관리자, 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계마다 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: PVC를 생성합니다.(`pvc1`) 대상 네임스페이스와 공유할 수 있는 권한을 부여하는 소스 네임스페이스에서(`namespace2`)를 사용하여 `shareToNamespace` 주석.

```
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 PV와 백엔드 NFS 스토리지 볼륨을 생성합니다.

- 쉼표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. 예를 들어, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` .
- 다음을 사용하여 모든 네임스페이스에 공유할 수 있습니다. `*` . 예를 들어, `trident.netapp.io/shareToNamespace: *`
- PVC를 업데이트하여 다음을 포함할 수 있습니다. `shareToNamespace` 언제든지 주석을 달 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하세요.
3. 대상 네임스페이스 소유자: 소스 네임스페이스를 참조하는 대상 네임스페이스에 TridentVolumeReference CR을 생성합니다. `pvc1` .

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

4. 대상 네임스페이스 소유자: PVC 생성(`pvc2`) 대상 네임스페이스에(`namespace2`)를 사용하여 `shareFromPVC` 소스 PVC를 지정하는 주석입니다.

```
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
```

대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

Trident 다음을 읽습니다. `shareFromPVC` 대상 PVC에 주석을 추가하고 자체 저장 리소스가 없는 종속 볼륨으로 대상 PV를 생성하여 소스 PV를 가리키고 소스 PV 저장 리소스를 공유합니다. 대상 PVC와 PV는 정상적으로 바인딩된 것으로 나타납니다.

공유 볼륨 삭제

여러 네임스페이스에서 공유되는 볼륨을 삭제할 수 있습니다. Trident 소스 네임스페이스에서 볼륨에 대한 액세스를 제거하고 볼륨을 공유하는 다른 네임스페이스에 대한 액세스를 유지합니다. 볼륨을 참조하는 모든 네임스페이스가 제거되면 Trident 볼륨을 삭제합니다.

사용 `tridentctl get` 하위 볼륨을 쿼리하려면

를 사용하여[`tridentctl` 유틸리티를 실행하면 됩니다 `get` 하위 볼륨을 가져오는 명령입니다. 자세한 내용은 링크를 참조하세요:../trident-reference/tridentctl.html[`tridentctl` 명령 및 옵션].

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

플래그:

- `-h, --help`: 볼륨에 대한 도움말.
- `--parentOfSubordinate string`: 하위 소스 볼륨에 대한 쿼리를 제한합니다.
- `--subordinateOf string`: 볼륨의 하위 항목으로 쿼리를 제한합니다.

제한 사항

- Trident 대상 네임스페이스가 공유 볼륨에 쓰는 것을 막을 수 없습니다. 공유 볼륨 데이터를 덮어쓰는 것을 방지하려면 파일 잠금이나 다른 프로세스를 사용해야 합니다.
- 소스 PVC에 대한 액세스를 제거하여 취소할 수 없습니다. `shareToNamespace` 또는 `shareFromNamespace` 주석이나 삭제 `TridentVolumeReference` 크.알. 접근 권한을 취소하려면 하위 PVC를 삭제해야 합니다.
- 하위 볼륨에서는 스냅샷, 복제 및 미러링이 불가능합니다.

더 많은 정보를 원하시면

네임스페이스 간 볼륨 액세스에 대해 자세히 알아보려면 다음을 참조하세요.

- 방문하다"네임스페이스 간 볼륨 공유: 네임스페이스 간 볼륨 액세스를 만나보세요" .
- 데모를 시청하세요"넷앱TV" .

네임스페이스 전체에서 볼륨 복제

Trident 사용하면 동일한 Kubernetes 클러스터 내의 다른 네임스페이스에서 기존 볼륨이나 볼륨 스냅샷을 사용하여 새 볼륨을 만들 수 있습니다.

필수 조건

볼륨을 복제하기 전에 소스 및 대상 백엔드가 동일한 유형이고 동일한 스토리지 클래스를 가지고 있는지 확인하세요.

> ⓘ 네임스페이스 간 복제는 다음에 대해서만 지원됩니다. `ontap-san` 그리고 `ontap-nas` 스토리지 드라이버. 읽기 전용 복제본은 지원되지 않습니다.

빠른 시작

몇 단계만 거치면 볼륨 복제를 설정할 수 있습니다.

1. 볼륨을 복제하기 위해 소스 **PVC**를 구성합니다.

소스 네임스페이스 소유자는 소스 PVC의 데이터에 액세스할 수 있는 권한을 부여합니다.

2. 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여합니다.

클러스터 관리자는 대상 네임스페이스 소유자에게 TridentVolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3. 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다.

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4. 대상 네임스페이스에 복제 **PVC**를 만듭니다.

대상 네임스페이스의 소유자는 소스 네임스페이스에서 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스 구성

보안을 보장하려면 네임스페이스 간에 볼륨을 복제하려면 소스 네임스페이스 소유자, 클러스터 관리자, 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계마다 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: PVC를 생성합니다.(`pvc1`) 소스 네임스페이스에서(`namespace1`) 대상 네임스페이스와 공유할 수 있는 권한을 부여합니다.(`namespace2`)를 사용하여 `cloneToNamespace` 주석.

```
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 PV와 백엔드 스토리지 볼륨을 생성합니다.

- 쉼표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. 예를 들어, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` .
- 다음을 사용하여 모든 네임스페이스에 공유할 수 있습니다. `*` . 예를 들어, `trident.netapp.io/cloneToNamespace: *`
- PVC를 업데이트하여 다음을 포함할 수 있습니다. `cloneToNamespace` 언제든지 주석을 달 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에서 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하십시오.(`namespace2`).
3. 대상 네임스페이스 소유자: 소스 네임스페이스를 참조하는 대상 네임스페이스에 TridentVolumeReference CR을 생성합니다. `pvc1` .

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

4. 대상 네임스페이스 소유자: PVC 생성(`pvc2`) 대상 네임스페이스에(`namespace2`)를 사용하여 `cloneFromPVC` 또는 `cloneFromSnapshot` , 그리고 `cloneFromNamespace` 소스 PVC를 지정하는 주석입니다.

```
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
```

제한 사항

- ontap-nas-economy 드라이버를 사용하여 프로비저닝된 PVC의 경우 읽기 전용 복제본은 지원되지 않습니다.

SnapMirror 사용하여 볼륨 복제

Trident 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 대상 볼륨 간의 미러 관계를 지원합니다. Trident Mirror Relationship(TMR)이라고 하는 네임스페이스가 지정된 사용자 지정 리소스 정의(CRD)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨(PVC) 간 미러 관계 생성
- 볼륨 간 미러 관계 제거
- 거울 속 관계를 깨세요

- 재해 상황(장애 조치) 동안 보조 볼륨을 홍보합니다.
- 계획된 장애 조치 또는 마이그레이션 중 클러스터 간에 애플리케이션의 손실 없는 전환을 수행합니다.

복제 전제 조건

시작하기 전에 다음 전제 조건이 충족되었는지 확인하세요.

ONTAP 클러스터

- * Trident *: ONTAP 백엔드로 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 Trident 버전 22.10 이상이 있어야 합니다.
- 라이선스: 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스는 소스 및 대상 ONTAP 클러스터 모두에서 활성화되어야 합니다. 참조하다 "ONTAP 의 SnapMirror 라이선싱 개요" 자세한 내용은.

  ONTAP 9.10.1부터 모든 라이선스는 여러 기능을 활성화하는 단일 파일인 NetApp 라이선스 파일(NLF)로 제공됩니다. 참조하다"ONTAP One에 포함된 라이센스" 자세한 내용은.

  SnapMirror 비동기 보호만 지원됩니다.

피어링

- 클러스터 및 **SVM**: ONTAP 스토리지 백엔드는 피어링되어야 합니다. 참조하다 "클러스터 및 SVM 피어링 개요" 자세한 내용은.

  두 ONTAP 클러스터 간 복제 관계에 사용된 SVM 이름이 고유한지 확인하세요.

- * Trident 및 SVM*: 피어링된 원격 SVM은 대상 클러스터의 Trident 에서 사용할 수 있어야 합니다.

지원되는 드라이버

NetApp Trident 다음 드라이버가 지원하는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 볼륨 복제를 지원합니다. **`ontap-nas`** : **NFS** `ontap-san` : iSCSI **`ontap-san`** : **FC** `ontap-san` : NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)

SnapMirror 사용한 볼륨 복제는 ASA r2 시스템에서는 지원되지 않습니다. ASA r2 시스템에 대한 정보는 다음을 참조하세요."ASA r2 스토리지 시스템에 대해 알아보세요" .

거울 **PVC** 만들기

다음 단계를 따르고 CRD 예제를 사용하여 기본 볼륨과 보조 볼륨 간의 미러 관계를 만듭니다.

단계

1. 기본 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. StorageClass 객체를 생성합니다. `trident.netapp.io/replication: true` 매개변수.

예

```
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. 이전에 생성한 StorageClass로 PVC를 생성합니다.

예

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

c. 지역 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

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

Trident 볼륨의 내부 정보와 볼륨의 현재 데이터 보호(DP) 상태를 가져온 다음 MirrorRelationship의 상태 필드를 채웁니다.

d. TridentMirrorRelationship CR을 가져와서 PVC의 내부 이름과 SVM을 얻습니다.

```
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. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개변수를 사용하여 StorageClass를 생성합니다.

      예

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

   b. 목적지 및 소스 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

```
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 구성된 관계 정책 이름(또는 ONTAP 의 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 보조(SnapMirror 대상) 역할을 하는 PVC를 생성합니다.

예

```
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 TridentMirrorRelationship CRD를 확인하고 해당 관계가 없으면 볼륨을 생성하지 못합니다. 관계가 존재하는 경우, Trident 새로운 FlexVol volume MirrorRelationship에 정의된 원격 SVM과 피어링된 SVM에 배치되도록 합니다.

볼륨 복제 상태

Trident 미러 관계(TMR)는 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 목적지 TMR에는 상태가 있는데, 이 상태는 Trident 원하는 상태가 무엇인지 알려줍니다. 목적지 TMR의 상태는 다음과 같습니다.

- 설립됨: 로컬 PVC는 미러 관계의 대상 볼륨이며, 이는 새로운 관계입니다.
- 홍보: 로컬 PVC는 읽기/쓰기가 가능하고 마운트 가능하며, 현재 미러 관계는 적용되지 않습니다.
- 재설정: 로컬 PVC가 미러 관계의 대상 볼륨이며 이전에도 해당 미러 관계에 있었습니다.

- 대상 볼륨이 소스 볼륨과 관계를 맺은 적이 있는 경우에는 재설정된 상태를 사용해야 합니다. 재설정된 상태는 대상 볼륨의 내용을 덮어쓰기 때문입니다.
- 볼륨이 이전에 소스와 관계가 없었던 경우 재설정된 상태는 실패합니다.

계획되지 않은 장애 조치 중 2차 PVC 홍보

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 *spec.state* 필드를 다음으로 업데이트합니다. `promoted` .

계획된 장애 조치 중 보조 PVC 홍보

계획된 장애 조치(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격합니다.

단계

1. 기본 Kubernetes 클러스터에서 PVC의 스냅샷을 만들고 스냅샷이 생성될 때까지 기다립니다.
2. 기본 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 얻습니다.

   예

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

3. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship* CR의 *spec.state* 필드를 _promoted_로 업데이트하고 _spec.promotedSnapshotHandle_을 스냅샷의 internalName으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 상태(status.state 필드)가 승격되었는지 확인합니다.

장애 조치 후 미러 관계 복원

미러 관계를 복원하기 전에 새로운 기본 관계로 만들려는 측면을 선택하세요.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.remoteVolumeHandle* 필드 값이 업데이트되었는지 확인하세요.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.mirror* 필드를 다음으로 업데이트합니다. `reestablished` .

추가 작업

Trident 기본 및 보조 볼륨에서 다음 작업을 지원합니다.

1차 **PVC**를 새로운 **2**차 **PVC**로 복제합니다**.**

기본 PVC와 보조 PVC가 이미 있는지 확인하세요.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 기본(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새로운 2차(대상) PVC에 대한 기본(소스) 클러스터에 새로운 TridentMirrorRelationship CRD를 만듭니다.

미러링된 **1**차 또는 **2**차 **PVC** 크기 조정

PVC는 일반적으로 크기를 조절할 수 있으며, 데이터 양이 현재 크기를 초과하면 ONTAP 모든 대상 flevxols를 자동으로 확장합니다.

PVC에서 복제 제거

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행합니다.

- 2차 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promoted_로 업데이트합니다.

PVC(이전에 미러링됨**)** 삭제

Trident 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR 삭제

미러링된 관계의 한쪽에서 TMR을 삭제하면 Trident 삭제를 완료하기 전에 나머지 TMR이 승격 상태로 전환됩니다. 삭제를 위해 선택된 TMR이 이미 *promoted* 상태인 경우 기존 미러 관계가 없으므로 TMR이 제거되고 Trident 로컬 PVC를 *ReadWrite*_로 승격합니다. 이 삭제로 *ONTAP* 의 로컬 볼륨에 대한 *SnapMirror* 메타데이터가 해제됩니다. 이 볼륨이 나중에 미러 관계에서 사용되는 경우 새 미러 관계를 만들 때 _설정된 볼륨 복제 상태를 가진 새 TMR을 사용해야 합니다.

ONTAP 이 온라인일 때 미러 관계 업데이트

미러 관계는 설정된 후 언제든지 업데이트할 수 있습니다. 당신은 사용할 수 있습니다 `state: promoted` 또는 `state: reestablished` 관계를 업데이트하는 필드입니다. 대상 볼륨을 일반 ReadWrite 볼륨으로 승격할 때 _promotedSnapshotHandle_을 사용하여 현재 볼륨을 복원할 특정 스냅샷을 지정할 수 있습니다.

ONTAP 이 오프라인일 때 미러 관계 업데이트

Trident ONTAP 클러스터에 직접 연결되지 않고도 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. TridentActionMirrorUpdate의 다음 예제 형식을 참조하세요.

예

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

`status.state`TridentActionMirrorUpdate CRD의 상태를 반영합니다. *Succeeded*, *In Progress*, 또는 *Failed* 값을 가질 수 있습니다.

CSI 토폴로지 사용

Trident Kubernetes 클러스터에 있는 노드에 볼륨을 선택적으로 생성하고 첨부할 수 있습니다. "CSI 토폴로지 기능" .

개요

CSI 토폴로지 기능을 사용하면 지역 및 가용성 영역을 기반으로 볼륨에 대한 액세스를 노드 하위 집합으로 제한할 수 있습니다. 오늘날 클라우드 공급업체는 Kubernetes 관리자가 영역 기반 노드를 생성할 수 있도록 지원합니다. 노드는 한 지역 내의 여러 가용성 영역에 위치할 수도 있고, 여러 지역에 걸쳐 위치할 수도 있습니다. 다중 구역 아키텍처의 워크로드에 대한 볼륨 프로비저닝을 용이하게 하기 위해 Trident CSI 토폴로지를 사용합니다.

CSI 토폴로지 기능에 대해 자세히 알아보세요 "여기" .

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- 와 함께 `VolumeBindingMode` 로 설정 `Immediate` Trident 토폴로지를 인식하지 않고 볼륨을 생성합니다. 볼륨 바인딩과 동적 프로비저닝은 PVC가 생성될 때 처리됩니다. 이것은 기본값입니다 `VolumeBindingMode` 토폴로지 제약을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청하는 포드의 스케줄링 요구 사항에 대한 종속성 없이 생성됩니다.
- 와 함께 `VolumeBindingMode` 로 설정 `WaitForFirstConsumer` PVC에 대한 영구 볼륨의 생성 및 바인딩은 PVC를 사용하는 포드가 예약되고 생성될 때까지 지연됩니다. 이런 방식으로 토폴로지 요구 사항에 의해 적용되는 스케줄링 제약 조건을 충족하는 볼륨이 생성됩니다.

그만큼 `WaitForFirstConsumer` 바인딩 모드에는 토폴로지 레이블이 필요하지 않습니다. 이 기능은 CSI 토폴로지 기능과 별도로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 활용하려면 다음이 필요합니다.

- Kubernetes 클러스터를 실행 중"지원되는 Kubernetes 버전"

```
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"}
```

- 클러스터의 노드에는 토폴로지 인식을 소개하는 레이블이 있어야 합니다 .(`topology.kubernetes.io/region` 그리고 `topology.kubernetes.io/zone`). Trident 가 토폴로지를 인식할 수 있도록 Trident 설치하기 전에 이러한 레이블이 클러스터의 노드에 있어야 합니다.

```
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"}]
```

1단계: 토폴로지 인식 백엔드 만들기

Trident 스토리지 백엔드는 가용성 영역에 따라 볼륨을 선택적으로 프로비저닝하도록 설계될 수 있습니다. 각 백엔드는 선택 사항을 수행할 수 있습니다. `supportedTopologies` 지원되는 영역 및 지역 목록을 나타내는 블록입니다. 이러한 백엔드를 사용하는 StorageClass의 경우, 지원되는 지역/영역에서 예약된 애플리케이션에서 요청하는 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

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`백엔드별 지역 및 영역 목록을 제공하는 데 사용됩니다. 이러한 지역과 영역은 StorageClass에서 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공되는 지역 및 영역의 하위 집합을 포함하는 StorageClass의 경우 Trident 백엔드에 볼륨을 생성합니다.

정의할 수 있습니다 `supportedTopologies` 저장 풀당도 마찬가지입니다. 다음 예를 참조하세요.

```
---
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
```

이 예에서는 `region` 그리고 `zone` 라벨은 저장 풀의 위치를 나타냅니다. `topology.kubernetes.io/region` 그리고 `topology.kubernetes.io/zone` 저장 풀을 어디에서 사용할 수 있는지 지정합니다.

2단계: 토폴로지를 인식하는 StorageClass 정의

클러스터의 노드에 제공된 토폴로지 레이블을 기반으로 토폴로지 정보를 포함하도록 StorageClass를 정의할 수 있습니다. 이를 통해 PVC 요청에 대한 후보 역할을 하는 스토리지 풀과 Trident 에서 프로비저닝한 볼륨을 활용할 수 있는 노드 하위 집합이 결정됩니다.

다음 예를 참조하세요.

```
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
```

위에 제공된 StorageClass 정의에서 `volumeBindingMode` 로 설정됩니다 `WaitForFirstConsumer` . 이 StorageClass로 요청된 PVC는 Pod에서 참조될 때까지 작업이 수행되지 않습니다. 그리고, `allowedTopologies` 사용할 구역과 지역을 제공합니다. 그만큼 `netapp-san-us-east1` StorageClass는 PVC를 생성합니다. `san-backend-us-east1` 백엔드는 위에 정의되어 있습니다.

3단계: PVC 만들기 및 사용

StorageClass를 생성하고 백엔드에 매핑했으므로 이제 PVC를 생성할 수 있습니다.

예를 보세요 `spec` 아래에:

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

이 매니페스트를 사용하여 PVC를 생성하면 다음과 같은 결과가 발생합니다.

```
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
```

Trident 볼륨을 생성하고 이를 PVC에 연결하려면 포드에서 PVC를 사용하세요. 다음 예를 참조하세요.

```
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
```

이 podSpec은 Kubernetes에 현재 존재하는 노드에서 Pod를 예약하도록 지시합니다. `us-east1` 지역 및 해당 지역에 있는 모든 노드 중에서 선택하십시오. `us-east1-a` 또는 `us-east1-b` 구역.

다음 출력을 확인하세요.

```
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
```

백엔드를 업데이트하여 포함합니다. `supportedTopologies`

기존 백엔드를 업데이트하여 다음 목록을 포함할 수 있습니다. `supportedTopologies` 사용 중 `tridentctl backend update` . 이는 이미 프로비저닝된 볼륨에는 영향을 미치지 않으며, 후속 PVC에만 사용됩니다.

더 많은 정보를 찾아보세요

- "컨테이너 리소스 관리"
- "노드 선택기"
- "친화성과 반친화성"
- "오염과 관용"

스냅샷 작업

영구 볼륨(PV)의 Kubernetes 볼륨 스냅샷을 사용하면 볼륨의 특정 시점 복사본을 만들 수 있습니다. Trident 사용하여 생성된 볼륨의 스냅샷을 생성하고, Trident 외부에서 생성된 스냅샷을 가져오고, 기존 스냅샷에서 새 볼륨을 생성하고, 스냅샷에서 볼륨 데이터를 복구할 수 있습니다.

개요

볼륨 스냅샷은 다음에서 지원됩니다. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `gcp-cvs` , `azure-netapp-files` , 그리고 `google-cloud-netapp-volumes` 운전자.

시작하기 전에

스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 사용자 정의 리소스 정의(CRD)가 필요합니다. 이는 Kubernetes 오케스트레이터(예: Kubeadm, GKE, OpenShift)의 책임입니다.

Kubernetes 배포판에 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요.볼륨 스냅샷 컨트롤러 배포 .

GKE 환경에서 주문형 볼륨 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마세요. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

볼륨 스냅샷 생성

단계

1. 생성하다 `VolumeSnapshotClass` 자세한 내용은 다음을 참조하세요."볼륨 스냅샷 클래스" .
   - 그만큼 `driver` Trident CSI 드라이버를 가리킨다.
   - `deletionPolicy`될 수 있다 `Delete` 또는 `Retain` . 설정 시 `Retain` , 스토리지 클러스터의 기본 물리적 스냅샷은 다음과 같은 경우에도 유지됩니다. `VolumeSnapshot` 객체가 삭제되었습니다.

   예

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

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

2. 기존 PVC의 스냅샷을 만듭니다.

   예시
   - 이 예제에서는 기존 PVC의 스냅샷을 만듭니다.

   ```
   cat snap.yaml
   ```

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

   - 이 예제에서는 PVC에 대한 볼륨 스냅샷 객체를 생성합니다. `pvc1` 그리고 스냅샷의 이름은 다음과 같이 설정됩니다. `pvc1-snap` . VolumeSnapshot은 PVC와 유사하며 다음과 연관됩니다. `VolumeSnapshotContent` 실제 스냅샷을 나타내는 객체입니다.

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

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

- 당신은 식별할 수 있습니다 `VolumeSnapshotContent` ~에 대한 객체 `pvc1-snap` VolumeSnapshot을 설명하여 보세요. 그만큼 `Snapshot Content Name` 이 스냅샷을 제공하는 VolumeSnapshotContent 객체를 식별합니다. 그만큼 `Ready To Use` 매개변수는 스냅샷을 사용하여 새로운 PVC를 생성할 수 있음을 나타냅니다.

```
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
...
```

볼륨 스냅샷에서 **PVC** 생성

사용할 수 있습니다 `dataSource` VolumeSnapshot이라는 이름을 사용하여 PVC를 생성하려면 `<pvc-name>` 데이터의 출처로서. PVC를 만든 후에는 포드에 부착하여 다른 PVC와 마찬가지로 사용할 수 있습니다.

PVC는 소스 볼륨과 동일한 백엔드에 생성됩니다. 참조하다"KB: Trident PVC 스냅샷에서 PVC를 생성하는 작업은 대체 백엔드에서 생성할 수 없습니다." .

다음 예제에서는 다음을 사용하여 PVC를 생성합니다. `pvc1-snap` 데이터 소스로.

```
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
```

볼륨 스냅샷 가져오기

Trident 다음을 지원합니다."Kubernetes 사전 프로비저닝 스냅샷 프로세스" 클러스터 관리자가 다음을 생성할 수 있도록 합니다. `VolumeSnapshotContent` Trident 외부에서 생성된 객체 및 가져오기 스냅샷.

시작하기 전에

Trident 스냅샷의 부모 볼륨을 생성하거나 가져와야 합니다.

단계

1. 클러스터 관리자: 생성 `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 개체입니다. 이렇게 하면 Trident 에서 스냅샷 워크플로가 시작됩니다.
   - 백엔드 스냅샷의 이름을 지정하세요. `annotations` ~처럼 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">` .
   - 지정하다 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` ~에 `snapshotHandle` . 이것은 외부 스냅샷터가 Trident 에 제공하는 유일한 정보입니다. `ListSnapshots` 부르다.

     그만큼 `<volumeSnapshotContentName>` CR 명명 제약으로 인해 백엔드 스냅샷 이름과 항상 일치할 수는 없습니다.

     예

     다음 예제에서는 다음을 생성합니다. `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 객체 `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. 클러스터 관리자: 생성 `VolumeSnapshot` 참조하는 CR `VolumeSnapshotContent` 물체. 이것은 사용에 대한 액세스를 요청합니다. `VolumeSnapshot` 주어진 네임스페이스에서.

   예

   다음 예제에서는 다음을 생성합니다. `VolumeSnapshot` CR이 명명됨 `import-snap` 참조하는 `VolumeSnapshotContent` 명명된 `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. 내부 처리(작업 필요 없음): 외부 스냅샷터는 새로 생성된 것을 인식합니다. `VolumeSnapshotContent` 그리고 실행됩니다 `ListSnapshots` 부르다. Trident 다음을 생성합니다. `TridentSnapshot` .
   - 외부 스냅샷터는 다음을 설정합니다. `VolumeSnapshotContent` 에게 `readyToUse` 그리고 `VolumeSnapshot` 에게 `true` .
   - Trident 돌아온다 `readyToUse=true` .
4. 모든 사용자: 만들기 `PersistentVolumeClaim` 새로운 것을 참조하기 위해 `VolumeSnapshot` , 여기서 `spec.dataSource` (또는 `spec.dataSourceRef`) 이름은 `VolumeSnapshot` 이름.

예

다음 예제에서는 다음을 참조하는 PVC를 생성합니다. `VolumeSnapshot` 명명된 `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
```

스냅샷을 사용하여 볼륨 데이터 복구

스냅샷 디렉토리는 기본적으로 숨겨져 있어 볼륨 프로비저닝의 최대 호환성을 용이하게 합니다. `ontap-nas` 그리고 `ontap-nas-economy` 운전자. 활성화 `.snapshot` 스냅샷에서 직접 데이터를 복구할 수 있는 디렉토리입니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

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

스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 적용된 변경 사항은 손실됩니다.

스냅샷에서 제자리 볼륨 복원

Trident 다음을 사용하여 스냅샷에서 빠르고 정확한 볼륨 복원을 제공합니다. `TridentActionSnapshotRestore` (TASR) CR. 이 CR은 필수 Kubernetes 작업으로 작동하며 작업이 완료된 후에는 유지되지 않습니다.

Trident 스냅샷 복원을 지원합니다. `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` , `google-cloud-netapp-volumes` , 그리고 `solidfire-san` 운전자.

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인하세요.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인하세요.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예제에서는 PVC에 대한 CR을 생성합니다. `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot` .

   TASR CR은 PVC 및 VS가 있는 네임스페이스에 있어야 합니다.

```
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. 스냅샷에서 복원하려면 CR을 적용합니다. 이 예제는 스냅샷에서 복원합니다. `pvc1` .

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

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

결과

Trident 스냅샷에서 데이터를 복원합니다. 스냅샷 복원 상태를 확인할 수 있습니다.

```
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: ""
```

- 대부분의 경우, Trident 실패 시 자동으로 작업을 다시 시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 액세스 권한이 없는 Kubernetes 사용자는 애플리케이션 네임스페이스에 TASR CR을 생성하기 위해 관리자로부터 권한을 부여받아야 할 수도 있습니다.

연관된 스냅샷이 있는 PV 삭제

연관된 스냅샷이 있는 영구 볼륨을 삭제하면 해당 Trident 볼륨이 "삭제 중 상태"로 업데이트됩니다. 볼륨 스냅샷을 제거하여 Trident 볼륨을 삭제합니다.

볼륨 스냅샷 컨트롤러 배포

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않으면 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
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. 스냅샷 컨트롤러를 생성합니다.

```
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
```

필요한 경우 열어주세요 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 그리고 업데이트 `namespace` 네임스페이스에.

관련 링크

- "볼륨 스냅샷"
- "볼륨 스냅샷 클래스"

볼륨 그룹 스냅샷 작업

영구 볼륨(PV)의 Kubernetes 볼륨 그룹 스냅샷 NetApp Trident 여러 볼륨(볼륨 스냅샷 그룹)의 스냅샷을 생성하는 기능을 제공합니다. 이 볼륨 그룹 스냅샷은 동일한 시점에 생성된 여러 볼륨의 복사본을 나타냅니다.

VolumeGroupSnapshot은 베타 API가 포함된 Kubernetes의 베타 기능입니다. VolumeGroupSnapshot에 필요한 최소 버전은 Kubernetes 1.32입니다.

볼륨 그룹 스냅샷 생성

볼륨 그룹 스냅샷은 다음과 같이 지원됩니다. `ontap-san` 드라이버는 iSCSI 프로토콜에만 적용되며, 아직 Fibre Channel(FCP)이나 NVMe/TCP에서는 지원되지 않습니다. .시작하기 전에

- Kubernetes 버전이 K8s 1.32 이상인지 확인하세요.
- 스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 사용자 정의 리소스 정의(CRD)가 필요합니다. 이는 Kubernetes 오케스트레이터(예: Kubeadm, GKE, OpenShift)의 책임입니다.

  Kubernetes 배포판에 외부 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요.볼륨 스냅샷 컨트롤러 배포 .

  > GKE 환경에서 주문형 볼륨 그룹 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마세요. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

- 스냅샷 컨트롤러 YAML에서 다음을 설정합니다. `CSIVolumeGroupSnapshot` 볼륨 그룹 스냅샷이 활성화되도록 기능 게이트를 'true'로 설정합니다.
- 볼륨 그룹 스냅샷을 생성하기 전에 필요한 볼륨 그룹 스냅샷 클래스를 생성하세요.
- VolumeGroupSnapshot을 생성하려면 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하세요.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하세요. 자세한 내용은 다음을 참조하세요."볼륨 그룹 스냅샷 클래스" .

```
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
```

- 기존 저장 클래스를 사용하여 필요한 라벨이 있는 PVC를 만들거나, 이러한 라벨을 기존 PVC에 추가합니다.

  다음 예제에서는 다음을 사용하여 PVC를 생성합니다. `pvc1-group-snap` 데이터 소스 및 레이블로 `consistentGroupSnapshot: groupA` . 요구 사항에 따라 레이블 키와 값을 정의합니다.

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

- 동일한 레이블로 VolumeGroupSnapshot을 만듭니다.(`consistentGroupSnapshot: groupA`) PVC에 지정됨.

  이 예제에서는 볼륨 그룹 스냅샷을 생성합니다.

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

그룹 스냅샷을 사용하여 볼륨 데이터 복구

볼륨 그룹 스냅샷의 일부로 생성된 개별 스냅샷을 사용하여 개별 영구 볼륨을 복원할 수 있습니다. 볼륨 그룹 스냅샷을 단위로 복구할 수 없습니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅샷에 기록된 상태로 복원합니다.

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

스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 적용된 변경 사항은 손실됩니다.

스냅샷에서 제자리 볼륨 복원

Trident 다음을 사용하여 스냅샷에서 빠르고 정확한 볼륨 복원을 제공합니다. `TridentActionSnapshotRestore` (TASR) CR. 이 CR은 필수 Kubernetes 작업으로 작동하며 작업이 완료된 후에는 유지되지 않습니다.

자세한 내용은 다음을 참조하세요. "스냅샷에서 제자리 볼륨 복원".

연관된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- 그룹의 개별 스냅샷이 아닌 VolumeGroupSnapshots 전체를 삭제할 수 있습니다.
- 스냅샷이 있는 동안 PersistentVolume이 삭제되면 Trident 해당 볼륨을 "삭제 중" 상태로 전환합니다. 볼륨을 안전하게 제거하기 전에 스냅샷을 제거해야 하기 때문입니다.
- 그룹화된 스냅샷을 사용하여 복제본을 만든 다음 그룹을 삭제하려는 경우 복제본 분할 작업이 시작되고 분할이 완료될 때까지 그룹을 삭제할 수 없습니다.

볼륨 스냅샷 컨트롤러 배포

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않으면 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
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. 스냅샷 컨트롤러를 생성합니다.

```
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
```

필요한 경우 열어주세요 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 그리고 업데이트 `namespace` 네임스페이스에.

관련 링크

- "볼륨 그룹 스냅샷 클래스"
- "볼륨 스냅샷"