볼륨 프로비저닝 및 관리

Trident

NetApp
July 01, 2026

목차

볼륨 프로비저닝 및 관리

볼륨 프로비저닝

구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim (PVC)를 생성합니다. 그런 다음 PV를 파드에 마운트할 수 있습니다.

개요

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolumeClaim_"^](PVC)`는 클러스터의 `PersistentVolume`에 대한 액세스 요청입니다.

PVC는 특정 크기의 스토리지 또는 액세스 모드를 요청하도록 구성할 수 있습니다. 연결된 StorageClass를 사용하여 클러스터 관리자는 PersistentVolume 크기 및 액세스 모드 외에도 성능이나 서비스 수준과 같은 다양한 요소를 제어할 수 있습니다.

PVC를 생성한 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 생성합니다

단계

1. PVC를 생성합니다.

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

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

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

1. POD에 볼륨을 마운트합니다.

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

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

2. 볼륨이 `/my/mount/path`에 마운트되었는지 확인하십시오.

```
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

이 예에서는 기본 PVC 구성 옵션을 보여 줍니다.

RWO 액세스가 있는 **PVC**

이 예제는 `basic-csi`라는 이름의 StorageClass와 연결된 RWO 액세스 권한이 있는 기본 PVC를 보여줍니다.

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

NVMe/TCP를 사용한 **PVC**

이 예제는 RWO 액세스가 가능한 NVMe/TCP용 기본 PVC를 보여주며, 이는 StorageClass라는 이름 `protection-gold`과 연결되어 있습니다.

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

Pod 매니페스트 샘플

이 예에서는 PVC를 포드에 연결하는 기본 구성을 보여 줍니다.

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
```

스토리지 클래스가 "Kubernetes 및 Trident 객체"와 상호 작용하는 방법 및 Trident가 볼륨을 프로비저닝하는 방식을 제어하는 매개변수에 대한 자세한 내용은 `PersistentVolumeClaim`를 참조하십시오.

볼륨 확장

Trident는 Kubernetes 사용자에게 볼륨 생성 후 확장 기능을 제공합니다. iSCSI, NFS, SMB, NVMe/TCP 및 FC 볼륨 확장에 필요한 구성에 대한 정보를 확인하십시오.

iSCSI 볼륨 확장

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

> iSCSI 볼륨 확장은 `ontap-san`, `ontap-san-economy`, `solidfire-san` 드라이버에서 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

StorageClass 정의를 편집하여 `allowVolumeExpansion` 필드를 `true`로 설정합니다.

```
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 `allowVolumeExpansion` 매개 변수를 포함하도록 편집합니다.

2단계: 생성한 **StorageClass**로 **PVC** 만들기

PVC 정의를 편집하고 새로 원하는 크기를 반영하도록 `spec.resources.requests.storage`을 업데이트합니다(원래 크기보다 커야 함).

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

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

Trident는 영구 볼륨(PV)을 생성하고 이 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 연결하는 **POD** 정의

크기를 조정할 PV를 포드에 연결합니다. iSCSI PV의 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 장치를 다시 스캔한 후 파일 시스템 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 파드에 바인딩된 후 Trident는 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

```
 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 `spec.resources.requests.storage`을 2Gi로 업데이트합니다.

```
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 `ontap-san` 드라이버에서 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

StorageClass 정의를 편집하여 `allowVolumeExpansion` 필드를 `true`로 설정합니다.

```
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 `allowVolumeExpansion` 매개 변수를 포함하도록 편집합니다.

2단계: 생성한 **StorageClass**로 **PVC** 만들기

PVC 정의를 편집하고 새로 원하는 크기를 반영하도록 `spec.resources.requests.storage`을 업데이트합니다(원래 크기보다 커야 함).

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

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

Trident는 영구 볼륨(PV)을 생성하고 이 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 연결하는 **POD** 정의

크기를 조정할 PV를 포드에 연결합니다. FC PV의 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 포드에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 장치를 다시 스캔한 후 파일 시스템 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 할 때 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 파드에 바인딩된 후 Trident는 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다. 그런 다음 Kubernetes는 확장 작업이 성공적으로 완료된 후 PVC 크기를 업데이트합니다.

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

```
 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 `spec.resources.requests.storage`을 2Gi로 업데이트합니다.

```
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

Trident는 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` 및 `azure-netapp-files` 백엔드에서 프로비저닝된 NFS PV의 볼륨 확장을 지원합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

NFS PV의 크기를 조정하려면 관리자가 먼저 `allowVolumeExpansion` 필드를 `true`로 설정하여 볼륨 확장을 허용하도록 스토리지 클래스를 구성해야 합니다:

```
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 `kubectl edit storageclass`을 사용하여 기존 스토리지 클래스를 편집하여 볼륨 확장을 허용할 수 있습니다.

2단계: 생성한 **StorageClass**로 **PVC** 만들기

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

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

Trident는 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 생성한 20 MiB PV의 크기를 1 GiB로 조정하려면 PVC를 편집하고 `spec.resources.requests.storage`을 1 GiB로 설정합니다:

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

RWX NVMe 하위 시스템 제한 사항 이해

ReadWriteMany(RWX) 볼륨은 NVMe 프로토콜을 사용하며 볼륨당 64개 노드의 확장성 제한이 있습니다. 다음은 제한 사항, 관련 NVMe 서브시스템 아키텍처 및 필요한 해결 단계를 설명합니다.

64개 노드 제한 이해

NVMe 프로토콜과 함께 ReadWriteMany(RWX) 볼륨을 사용하려는 경우, 단일 RWX NVMe 볼륨은 Kubernetes 클러스터에서 64개 이상의 노드에 마운트할 수 없습니다.

64개 이상의 노드에 걸쳐 동일한 RWX NVMe PersistentVolumeClaim을 마운트하는 워크로드를 예약하지 마십시오.

이 제한 사항은 NVMe 프로토콜을 사용하는 RWX 볼륨에만 적용됩니다.

NVMe 하위 시스템 모델 이해

볼륨별 하위 시스템 모델**(Trident 26.02** 이전 릴리스**)**

Trident 26.02 이전 릴리스에서는 RWX NVMe 볼륨이 볼륨별 서브시스템 모델을 사용하여 프로비저닝됩니다. 각 RWX NVMe 볼륨은 ONTAP의 자체 전용 NVMe 서브시스템에 매핑됩니다.

이 모델은 단순하지만 확장성 한계가 낮습니다. 대규모 Kubernetes 클러스터에서는 각 RWX 볼륨이 전용 서브시스템을 사용하기 때문에 서브시스템 컨트롤러 한계에 빠르게 도달합니다.

슈퍼 서브시스템 모델**(Trident 26.02**에서 도입**)**

Trident 26.02부터 RWX NVMe 볼륨은 공유 슈퍼 서브시스템 모델을 사용합니다. 여러 RWX NVMe 볼륨이 동일한 NVMe 서브시스템을 공유합니다.

각 슈퍼 서브시스템은 최대 1024개의 네임스페이스(볼륨)를 지원합니다. 이 모델은 RWX 워크로드의 확장성을 크게 향상시키고 ONTAP 서브시스템 한계에 도달할 가능성을 줄입니다.

각 RWX NVMe 볼륨은 최대 64개의 노드를 지원합니다.

오류 증상 파악

대규모로 RWX NVMe 볼륨을 생성하거나 연결하는 경우 다음과 유사한 오류가 발생할 수 있습니다.

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

이 오류는 ONTAP NVMe 서브시스템 컨트롤러 제한에 도달했음을 나타냅니다.

하위 시스템 제한 오류 해결

볼륨별 하위 시스템 제한을 넘어 슈퍼 하위 시스템 모델을 활용하려면 Trident 26.02 이상으로 업그레이드하십시오.

Trident를 업그레이드하여 슈퍼 서브시스템 모델을 적용합니다

RWX NVMe 볼륨에 슈퍼 서브시스템 모델을 적용하려면 다음을 수행합니다.

1. Trident를 버전 26.02 이상으로 업그레이드하십시오.
2. RWX NVMe 볼륨을 사용하는 모든 파드를 0개의 복제본으로 축소합니다.
3. RWX NVMe 볼륨을 적극적으로 사용하는 워크로드가 없는지 확인하십시오.
4. 포드를 다시 확장합니다.

이 재시작 시퀀스는 RWX NVMe 볼륨이 super-subsystem 모델을 사용하여 연결되도록 보장합니다.

- 이 제한 사항은 NVMe 프로토콜을 사용하는 RWX 볼륨에만 적용됩니다.
- 64노드 제한은 RWX NVMe 볼륨당 적용됩니다.
- 다른 액세스 모드 및 다른 프로토콜은 영향을 받지 않습니다.

컨트롤러 확장성

Trident는 여러 스토리지 드라이버에 걸쳐 향상된 동시성을 통해 컨트롤러 확장성을 제공합니다. 고객은 일반 출시 시점에 컨트롤러 확장성을 지원하는 Trident 드라이버와 Trident 26.02에서 기술 미리 보기로 제공되는 드라이버를 확인할 수 있습니다. 이를 통해 확장 가능한 Kubernetes 환경을 위한 정보에 기반한 배포 결정을 내리고 적절한 위험 관리를 수행할 수 있습니다.

핵심 개념 및 정의

컨트롤러 확장성

컨트롤러 확장성은 Trident 컨트롤러가 단일 잠금 뒤에서 여러 스토리지 작업을 직렬화하는 대신 병렬로 처리할 수 있는 능력을 의미합니다. 이러한 작업에는 볼륨 생성, 삭제, 크기 조정, 스냅샷 생성 및 삭제, 볼륨 게시 및 게시 취소, 백엔드 관리가 포함됩니다.

컨트롤러 확장성이 활성화되면 서로 다른 볼륨 및 백엔드에 대한 작업이 동시에 진행됩니다. 이는 PersistentVolumeClaim 및 VolumeSnapshot 작업이 동시에 많이 발생하는 환경에서 처리량을 높이고 엔드 투 엔드 작업 시간을 단축합니다.

컨트롤러 확장성 지원

Trident는 스토리지 드라이버에 따라 다양한 성숙도 수준의 컨트롤러 확장성을 지원합니다.

일반 가용성

다음 드라이버는 Trident 26.02의 정식 출시 버전에서 컨트롤러 확장성을 지원합니다.

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

> `google-cloud-netapp-volumes`와 `google-cloud-netapp-volumes-san` 드라이버는 서로 다릅니다. `google-cloud-netapp-volumes`만 지원됩니다. 백엔드 구성이나 예제에서 `google-cloud-netapp-volumes-san`를 사용하지 마십시오.

컨트롤러 확장성 활성화

컨트롤러 확장성은 `enableConcurrency` 구성 옵션을 통해 제어됩니다. 이 옵션은 Trident 설치 중 또는 기존 배포 업데이트 시 명시적으로 활성화해야 합니다.

Trident 운영자 배포

Trident operator에서 컨트롤러 확장성을 활성화하려면, `enableConcurrency`를 `true`로 `TridentOrchestrator` 커스텀 리소스에서 설정하세요.

신규 설치

`TridentOrchestrator` CR을 생성하거나 편집하고 `enableConcurrency`을(를) `true`(으)로 설정합니다:

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

CR 적용:

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

기존 설치

기존 `TridentOrchestrator` CR에 패치를 적용하여 컨트롤러 확장성을 활성화합니다.

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

설정이 적용되었는지 확인합니다.

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

Helm 배포

Helm을 사용하여 컨트롤러 확장성을 활성화하려면 `enableConcurrency` 값을 `true`로 설정하십시오.

신규 설치

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

기존 설치

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

또는, 사용자 지정 `values.yaml` 파일에서 `enableConcurrency`을(를) `true`(으)로 설정하십시오:

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

그런 다음 값 파일을 사용하여 설치 또는 업그레이드하십시오.

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

tridentctl 배포

`tridentctl`을(를) 사용하여 컨트롤러 확장성을 활성화하려면 설치 중에 `--enable-concurrency` 플래그를 전달하십시오.

신규 설치

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

기존 설치

기존 `tridentctl` 기반 배포에서 컨트롤러 확장성을 활성화하려면 다음 플래그를 사용하여 제거한 후 다시 설치하십시오.

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

컨트롤러 확장성이 활성화되어 있는지 확인하십시오

컨트롤러 확장성을 활성화한 후 컨트롤러 파드 로그를 확인하여 Trident 컨트롤러가 동시 실행 기능을 활성화한 상태로 실행 중인지 확인하십시오.

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

동시 실행이 활성화되었음을 나타내는 로그 항목이 표시됩니다.

기술 미리 보기

다음 드라이버는 Trident 26.02에서 기술 미리 보기로 컨트롤러 확장성을 지원합니다.

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

다음 드라이버의 경우:

- 컨트롤러 동시성을 평가 및 테스트할 수 있습니다.
- 향후 릴리스에서 동작 방식이 변경될 수 있습니다.
- 운영 환경에서는 사용하지 않는 것이 좋습니다

동시성 동작

컨트롤러 확장성이 활성화된 경우:

- Trident는 단일 전역 잠금을 리소스별 세분화된 잠금으로 대체합니다.
- 동일한 리소스를 수정하는 작업은 데이터 일관성을 유지하기 위해 직렬화됩니다
- 리소스에서 읽기만 하는 작업은 해당 리소스에 대한 다른 읽기 작업과 동시에 진행될 수 있습니다.
- Trident는 백엔드 스토리지 시스템의 과부하를 방지하기 위해 관리 LIF당 동시 ONTAP API 요청을 20개로 제한합니다
- 여러 백엔드가 동일한 관리 LIF를 공유하는 경우 이 20개 요청 제한을 공유합니다

알려진 제한 사항 및 고려 사항

컨트롤러 확장성에는 다음 사항이 적용됩니다.

- 동시성은 Trident 컨트롤러에 의해 내부적으로 관리됩니다
- 이번 릴리스에는 사용자가 구성할 수 있는 동시성 제한이 없습니다
- 전체 처리량은 다음 요소에 따라 달라집니다.
  - 사용 중인 스토리지 드라이버
  - 백엔드 응답성
  - Kubernetes API 서버 성능
- 높은 동시성은 백엔드 스토리지 시스템의 부하를 증가시킬 수 있습니다

주의사항 및 제한사항

Trident 26.02에는 다음과 같은 제한 사항이 적용됩니다.

- 컨트롤러 확장성 동작은 모든 드라이버에서 동일하지 않습니다
- 기술 미리 보기 드라이버에서 다음과 같은 현상이 나타날 수 있습니다.
  - 높은 부하에서 일관되지 않은 성능

    - 릴리스 간 동작 변경 사항
- 병렬 실행으로 인해 동시 작업 디버깅이 더 복잡해질 수 있습니다.
- 지표 및 로그에는 인터리브된 작업 출력이 표시될 수 있습니다

권장 사항

- 높은 확장성이 필요한 운영 환경에는 일반 가용성(GA) 드라이버를 사용하십시오
- 비프로덕션 환경에서 기술 미리 보기 드라이버를 평가합니다
- 대규모로 운영할 때 백엔드 및 컨트롤러 성능 모니터링
- 자동화 스크립트에서 작업 순서를 가정하지 마십시오

자동 볼륨 확장

자동 볼륨 확장을 사용하면 Trident에서 프로비저닝한 영구 볼륨이 사용된 용량이 정의된 임계값에 도달할 때 자동으로 증가합니다. 이 기능은 운영 오버헤드를 줄이고 용량 고갈로 인한 애플리케이션 중단을 방지하는 데 도움이 됩니다. 자동 볼륨 확장은 Autogrow 정책을 사용하여 구현됩니다. Autogrow 정책은 다음을 정의합니다.

- 확장을 트리거하는 사용률 임계값
- 볼륨이 증가하는 양
- 볼륨이 도달할 수 있는 최대 크기

> 정의된 사용률 임계값을 초과하면 볼륨 크기가 자동으로 증가합니다. 볼륨은 자동으로 축소되지 않습니다.

요구 사항

자동 볼륨 확장을 구성하기 전에 다음 요구 사항을 충족하는지 확인하십시오.

- Trident 26.02 이상
- 역할 기반 액세스 제어 권한으로 `TridentAutogrowPolicy` 사용자 지정 리소스 생성
- StorageClasses로 구성되었습니다 `allowVolumeExpansion: true`
- 지원되는 ONTAP 프로토콜:
    - 네트워크 파일 시스템(NFS)
    - Internet Small Computer Systems Interface(iSCSI)
    - 비휘발성 메모리 익스프레스(NVMe)
    - 파이버 채널 프로토콜(FCP)

제한 사항

- ONTAP 9.16.1 이전 버전의 ONTAP Non-Volatile Memory Express 원시 블록 볼륨은 자동 확장을 지원하지 않습니다.

- 스토리지 영역 네트워크 볼륨의 경우, `growthAmount`이(가) 50메비바이트 이하이면 Trident는 크기 조정 전에 자동으로 값을 51메비바이트로 늘립니다. 단, 결과 크기가 `maxSize`을(를) 초과하지 않는 경우에 한합니다.
- 브라운필드 환경에서는 볼륨 게시 마이그레이션 동작으로 인해 특정 기존 볼륨에 대해 자동 확장이 작동하지 않을 수 있습니다.
- 볼륨이 `maxSize`에 도달하면 더 이상 확장되지 않습니다.
- 자동 볼륨 확장을 위해 지원되는 프로토콜:
  - 네트워크 파일 시스템(NFS)
  - Internet Small Computer Systems Interface(iSCSI)
  - 비휘발성 메모리 익스프레스(NVMe)
  - 파이버 채널 프로토콜(FCP)

자동 확장 정책이 적용된 볼륨 프로비저닝

자동 확장 정책은 두 가지 수준에서 구성할 수 있습니다.

- 스토리지 클래스 수준: 모든 볼륨의 기본값을 설정합니다(주석 사용)
- PVC 레벨: 스토리지 클래스 기본값을 재정의합니다(annotation 사용)

자동 증가 정책 생성

자동 확장 정책을 사용하면 볼륨이 정의된 용량 임계값에 도달할 때 볼륨이 자동으로 확장됩니다.

다음 사항을 확인하십시오.

- Trident 26.02 이상이 설치되어 있습니다
- 역할 기반 액세스 제어 권한을 통해 `TridentAutogrowPolicy` 리소스를 생성합니다
- 워크로드 증가 요구 사항에 대한 이해

자동 확장 정책은 볼륨이 정의된 용량 임계값에 도달했을 때 자동으로 확장되는 방식을 정의합니다.

워크플로의 어느 단계에서든 자동 확장 정책을 생성할 수 있습니다.

- StorageClasses와 볼륨이 생성되기 전에
- StorageClasses가 존재한 후
- 볼륨이 프로비저닝된 후

이러한 유연성을 통해 기존 리소스를 다시 생성하지 않고도 자동 확장을 도입할 수 있습니다.

자동 확장 정책 사양

자동 증가 정책은 다음과 같이 정의된 Kubernetes 사용자 지정 리소스입니다.

필드	설명	형식	필수	예	기본값
이름	고유 정책 식별자	문자열	예	production-db-policy	None

필드	설명	형식	필수	예	기본값
usedThreshold	확장을 트리거하는 용량 비율	백분율 문자열	예	"80%"	None
growthAmount	임계값에 도달했을 때 증가할 양	백분율 또는 크기	아니요	"10%" 또는 "5Gi"	"10%"
maxSize	최대 볼륨 크기 제한	Kubernetes 수량	아니요	"500Gi"	무제한

자동 증가 정책 생성

단계

1. Autogrow Policy를 정의하는 YAML 파일을 생성합니다.

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

2. 클러스터에 정책을 적용합니다.

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

3. 정책이 생성되었는지 확인하십시오.

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

예상 출력

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

정책 상태

정책을 생성하면 Trident가 사양을 검증하고 다음 상태 중 하나를 할당합니다.

상태	설명	조치 필요
성공	정책이 검증되어 사용할 준비가 되었습니다.	없음.

상태	설명	조치 필요
실패	유효성 검사 오류가 감지되었습니다.	사양을 검토하고 수정하십시오.
삭제 중	삭제가 진행 중입니다.	완료될 때까지 기다립니다.

정책을 StorageClass에 연결합니다

`trident.netapp.io/autogrowPolicy` 어노테이션을 사용하여 StorageClass에 자동 확장 정책을 연결할 수 있습니다. 해당 StorageClass에서 프로비저닝된 모든 볼륨은 이 정책을 상속받습니다.

StorageClass는 다음을 가져야 합니다 `allowVolumeExpansion: true`.

단계

1. Autogrow Policy 어노테이션을 사용하여 StorageClass를 생성하거나 수정합니다.

```
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. StorageClass를 적용합니다.

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

3. 주석을 확인하세요:

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

예상 출력

```
production-db-policy
```

정책 우선 순위

StorageClass와 PVC 모두에 Autogrow Policy 어노테이션이 설정된 경우 Trident는 다음 우선순위 규칙을 적용합니다.

1. **PVC** 주석이 우선합니다**.** PVC에 `trident.netapp.io/autogrowPolicy`가 설정되면 해당 값이 항상 사용됩니다.
2. **StorageClass** 어노테이션은 **PVC**에 어노테이션이 없는 경우에만 적용됩니다**.**
3. 어노테이션이 둘 다 없는 경우 Autogrow Policy가 적용되지 않습니다.

StorageClass 어노테이션	**PVC** 주석	효과적인 행동
`trident.netapp.io/autogrowPolicy: standard-agp`	설정되지 않음	용도 `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	`logs-policy` 사용(PVC가 StorageClass를 재정의함).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	자동 증가 정책 없음(PVC에서 자동 증가 기능을 비활성화함).
설정되지 않음	`trident.netapp.io/autogrowPolicy: dev-policy`	용도 `dev-policy`.
설정되지 않음	설정되지 않음	자동 증가 정책 없음.

구성 예

다음 예시는 다양한 사용 사례에 대한 일반적인 Autogrow Policy 구성을 보여줍니다.

운영 데이터베이스에 대한 보수적인 정책

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

고정 증가 단위로 로그 스토리지

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

기본값이 포함된 최소 정책

`growthAmount`및 `maxSize`를 생략하면 Trident는 기본값(`10%` 증가, 무제한 크기)을 사용합니다.

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

사용자 지정 **maxSize** 및 기본 **growthAmount**가 있는 정책

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

무제한 **maxSize**로 공격적인 성장

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

소수 백분율이 포함된 정책

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

소수점 이하 백분율도 지원됩니다. 소수점 이하 세 자리 이상을 지정하면 Trident가 값을 소수점 셋째 자리까지 반올림합니다.

자동 확장 기능이 있는 **NAS StorageClass**

```
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
```

자동 확장 기능이 있는 **SAN StorageClass**

```
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
```

자동 증가 정책 관리

Autogrow 정책을 생성한 후에는 필요에 따라 정책을 보고, 업데이트하고, 삭제할 수 있습니다. 또한 특정 정책을 사용 중인 볼륨을 모니터링할 수도 있습니다.

자동 증가 정책 보기

모든 정책 나열

`kubectl`를 사용하여 클러스터의 모든 `Autogrow` 정책을 나열합니다.

```
kubectl get tridentautogrowpolicy
```

또는 다음을 사용하십시오 `tridentctl`:

```
tridentctl get autogrowpolicy
```

정책 세부 정보 보기

정책의 전체 사양 및 상태를 보려면 다음을 수행합니다.

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

YAML 형식으로 정책과 해당 정책에 연결된 볼륨을 보려면 다음을 수행합니다.

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

자동 증가 정책 업데이트

기존 정책을 수정하여 임계값, 증가량 또는 최대 크기를 변경할 수 있습니다. 변경 사항은 해당 정책을 사용하는 모든 볼륨에 즉시 적용됩니다.

> ⓘ 변경 사항은 현재 해당 정책을 사용 중인 모든 볼륨에 적용됩니다. 가능하면 먼저 비운영 환경에서 변경 사항을 테스트하십시오.

단계

1. 정책을 편집합니다.

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

2. `spec` 필드를 필요에 따라 수정합니다.

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

3. 저장 후 종료합니다. 변경 사항은 즉시 적용됩니다.

업데이트 고려 사항

- 즉시 효력 발생: 해당 정책을 사용하는 모든 볼륨은 다음 증가 평가 시 새로운 매개변수를 적용받습니다.
- 볼륨 재시작이 필요 없습니다. 변경 사항은 다음 확장 작업에 적용됩니다.
- 먼저 테스트: 가능하면 비운영 환경에서 변경 사항을 검증하십시오.
- 변경 사항 전달: 공유 정책을 수정할 때 팀에 알립니다.

자동 증가 정책 삭제

자동 확장 정책은 볼륨이 활발하게 사용 중인 동안 실수로 삭제되는 것을 방지하기 위해 finalizer 보호 기능을 사용합니다.

단계

1. 정책을 삭제합니다.

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

2. 볼륨이 여전히 정책을 사용 중인 경우 삭제 작업이 `Deleting` 상태로 전환됩니다. 어떤 볼륨이 영향을 받는지 확인하십시오:

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

3. 영향을 받는 각 볼륨에서 정책을 제거합니다. 다음 옵션 중 하나를 선택하십시오.
   - 옵션 **A:** 주석을 `**"none"**`로 설정하여 자동 크기 증가를 명시적으로 비활성화:

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

   - 옵션 **B:** 주석을 완전히 제거합니다.

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

삭제 동작

시나리오	동작
해당 정책을 사용하는 볼륨은 없습니다	정책은 즉시 삭제됩니다.
볼륨이 정책을 사용하고 있습니다	정책이 `Deleting` 상태에 진입합니다. 최종화 도구는 모든 볼륨이 제거될 때까지 완료를 차단합니다.
모든 볼륨이 정책에서 제거됩니다	최종 결정자가 제거되고 정책이 삭제됩니다.

자동 증가 정책 사용량 모니터링

정책을 사용하여 볼륨 확인

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

볼륨이 사용하는 정책 찾기

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

정책 이벤트 모니터링

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

지원되는 프로토콜

Autogrow는 다음과 같은 스토리지 프로토콜을 지원합니다:

- NFS
- iSCSI
- FCP
- NVMe

SAN 볼륨의 경우, 구성된 `growthAmount`이 50MiB 이하이면 Trident는 크기 조정 작업 시 증가량을 자동으로 51MB로 늘립니다. 단, 최종 크기가 `maxSize`를 초과하지 않는 경우에 한합니다.

알려진 제한 사항

- * ONTAP NVMe 원시 블록 볼륨:* ONTAP 9.16.1 이전 버전으로 생성된 볼륨은 자동 확장을 지원하지 않습니다.
- 기존 볼륨(브라운필드 배포): 유효한 Autogrow Policy가 적용되었더라도 기존 볼륨에서는 Autogrow 기능이 작동하지 않을 수 있습니다. 이는 볼륨 게시 마이그레이션이 진행 중이기 때문입니다. 마이그레이션이 완료되었는지 확인하려면 Trident 컨트롤러 로그에서 "`Migration completed`" 메시지를 확인하십시오.

자주 묻는 질문

Trident는 언제 임계값을 평가합니까**?**

Trident는 볼륨 사용량을 지속적으로 모니터링합니다. 사용 용량이 `usedThreshold`를 초과하면 Trident는 내부 크기 조정 요청을 생성하고 구성된 `growthAmount`만큼 볼륨을 확장합니다.

예를 들어, 이 정책은 용량의 80%에 도달하면 확장을 시작하고 매번 10%씩 증가시켜 최대 500 GiB까지 늘립니다.

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

볼륨이 이미 프로비저닝된 후에도 정책을 적용할 수 있습니까**?**

예. 자동 증가 정책은 언제든지 생성할 수 있으며, `trident.netapp.io/autogrowPolicy` 어노테이션을 추가하거나 업데이트하여 기존 PVC에 적용할 수 있습니다. PVC나 StorageClass를 다시 생성할 필요가 없습니다.

기존 PVC에 정책을 적용하려면:

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

기존 StorageClass에 정책을 적용하려면 다음을 수행합니다.

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

StorageClass와 PVC 모두에 자동 증가 정책을 설정하면 어떻게 되나요?

PVC 어노테이션이 항상 우선합니다. PVC에 `trident.netapp.io/autogrowPolicy` 어노테이션이 있는 경우 Trident는 StorageClass에서 지정한 값과 관계없이 해당 값을 사용합니다. 자세한 내용은 "정책 우선 순위"를 참조하십시오.

예를 들어, 다음과 같은 StorageClass가 있습니다.

```
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
```

그리고 StorageClass 정책을 재정의하는 이 PVC는 다음과 같습니다.

```
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
```

Trident는 `logs-policy`를 위해 `database-pvc`를 사용하며, `standard-agp`는 사용하지 않습니다.

특정 볼륨의 자동 확장을 비활성화하려면 어떻게 해야 하나요?

PVC 주석을 `"none"`로 설정합니다. 이렇게 하면 해당 볼륨에 대한 StorageClass 수준 정책이 재정의됩니다.

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

자동 크기 증가 기능이 비활성화되어 있는지 확인할 수 있습니다.

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

예상 출력

```
none
```

볼륨이 **maxSize**에 도달하면 어떻게 됩니까**?**

Trident가 볼륨 확장을 중지합니다. 사용량이 `usedThreshold`를 초과하여 계속 증가하더라도 해당 볼륨에 대한 추가 크기 조정 요청은 생성되지 않습니다.

예를 들어, 이 정책을 사용하면 Trident는 볼륨이 100GiB에 도달하면 볼륨 증가를 중지합니다.

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

무제한 성장을 허용하려면 `maxSize`을 생략하거나 `0`로 설정하십시오:

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

볼륨을 재시작하지 않고 정책을 변경할 수 있습니까**?**

예. 정책을 업데이트하면 해당 정책을 사용하는 모든 볼륨이 다음 용량 증가 평가 시 새 매개변수를 적용합니다. 볼륨을 다시 시작할 필요는 없습니다.

정책을 업데이트하려면 다음을 수행합니다.

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

필요에 따라 필드를 수정합니다.

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

저장 후 종료합니다. 업데이트된 정책을 확인합니다.

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

예상 출력

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

내 정책이 실패 상태인 이유는 무엇입니까?

`Failed` 상태는 정책 사양에 유효성 검사 오류가 있음을 나타냅니다. 다음 명령을 실행하여 오류 세부 정보를 확인하십시오.

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

일반적인 원인으로는 유효하지 않은 `usedThreshold`(1~99%여야 함), `growthAmount`을 초과하는 `maxSize` 또는 유효하지 않은 Kubernetes 수량 형식이 있습니다. 사양을 수정하고 다시 적용하십시오:

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

정책을 삭제할 수 없는 이유는 무엇인가요?

정책은 finalizer 보호 기능을 사용합니다. 볼륨이 여전히 정책을 사용 중인 경우 삭제 작업은 `Deleting` 상태로 들어가며 모든 볼륨이 정책에서 제거될 때까지 대기합니다.

영향을 받는 볼륨을 식별하십시오.

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

그런 다음 각 PVC에서 주석을 제거합니다.

```
# 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-
```

모든 볼륨이 제거되면 파이널라이저가 해제되고 정책이 삭제됩니다.

자동 크기 증가 기능은 모든 **ONTAP** 백엔드에서 작동하나요?

자동 확장 기능은 NFS, iSCSI, FCP 및 NVMe 프로토콜을 지원합니다. 단, NVMe 원시 블록 볼륨을 사용하려면 ONTAP 9.16.1 이상 버전이 필요합니다.

기존 배포 환경의 경우 자동 확장이 적용되기 전에 볼륨 게시 마이그레이션이 완료되어야 할 수 있습니다. Trident 컨트롤러 로그를 확인하여 마이그레이션 상태를 확인하십시오.

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

다음 StorageClass 예제는 NAS 및 SAN 백엔드에 대해 자동 확장이 구성된 방식을 보여줍니다.

NAS 백엔드

```
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 백엔드

```
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
```

SAN 볼륨의 최소 증가량은 얼마입니까?

SAN 볼륨의 경우, 유효 최소 증가량은 51MB입니다. `growthAmount`를 50MiB 이하로 설정한 경우, Trident는 크기 조정 작업 시 증가량을 자동으로 51MB로 늘립니다.

예를 들어, 이 정책은 `growthAmount`을 `"40Mi"`로 설정하지만, Trident는 이 정책을 사용하는 모든 SAN 볼륨에 대해 51MB의 증가분을 적용합니다.

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

이러한 자동 조정을 방지하려면 `growthAmount`을 50 MiB보다 큰 값으로 설정하십시오.

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

볼륨 가져오기

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

다음과 같은 경우 Trident로 볼륨을 가져올 수 있습니다.

- 애플리케이션을 컨테이너화하고 기존 데이터 세트를 재사용합니다
- 임시 애플리케이션에 데이터 세트의 클론 사용
- 실패한 Kubernetes 클러스터 재구축
- 재해 복구 중 애플리케이션 데이터 마이그레이션

고려 사항

볼륨을 가져오기 전에 다음 사항을 검토하십시오.

- Trident는 RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 타겟 볼륨입니다. 볼륨을 Trident로 가져오기 전에 미러 관계를 끊어야 합니다.
- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 활성 상태인 볼륨을 가져오려면 볼륨 클론을 생성한 다음 가져오기를 수행하십시오.

  이는 특히 블록 볼륨의 경우 중요한데, Kubernetes가 이전 연결을 인식하지 못하고 활성 볼륨을 Pod에 연결할 수 있기 때문입니다. 이로 인해 데이터 손상이 발생할 수 있습니다.

- `StorageClass`PVC에 지정되어야 하지만 Trident는 가져오기 중에 이 매개 변수를 사용하지 않습니다. 스토리지 클래스는 볼륨 생성 중에 스토리지 특성을 기반으로 사용 가능한 풀에서 선택하는 데 사용됩니다. 볼륨이 이미 존재하므로 가져오기 중에 풀 선택이 필요하지 않습니다. 따라서 볼륨이 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드 또는 풀에 존재하더라도 가져오기가 실패하지 않습니다.
- 기존 볼륨 크기는 PVC에서 확인 및 설정됩니다. 스토리지 드라이버가 볼륨을 가져온 후 PVC에 대한 ClaimRef를 포함하는 PV가 생성됩니다.
  - 재확보 정책은 처음에 PV에서 `retain`로 설정됩니다. Kubernetes가 PVC와 PV를 성공적으로 바인딩한 후 재확보 정책이 스토리지 클래스의 재확보 정책과 일치하도록 업데이트됩니다.
  - 스토리지 클래스의 회수 정책이 `delete`인 경우, PV가 삭제될 때 스토리지 볼륨도 삭제됩니다.
- 기본적으로 Trident는 PVC를 관리하고 백엔드에서 FlexVol 볼륨과 LUN의 이름을 변경합니다. 관리되지 않는 볼륨을 가져오려면 `--no-manage` 플래그를, 볼륨 이름을 유지하려면 `--no-rename` 플래그를 전달할 수 있습니다.
  - `--no-manage`* - `--no-manage` 플래그를 사용하면 Trident는 객체의 수명 주기 동안 PVC 또는 PV에 대해 추가 작업을 수행하지 않습니다. PV가 삭제될 때 스토리지 볼륨은 삭제되지 않으며 볼륨 클론 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다.
  - `--no-rename`* - `--no-rename` 플래그를 사용하면 Trident는 볼륨을 가져오는 동안 기존 볼륨 이름을 유지하고 볼륨의 수명 주기를 관리합니다. 이 옵션은 `ontap-nas`, `ontap-san`(ASA r2 시스템 포함) 및 `ontap-san-economy` 드라이버에서만 지원됩니다.

이러한 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하면서도 스토리지 볼륨의 수명 주기는 Kubernetes 외부에서 관리하려는 경우에 유용합니다.

- PVC 및 PV에는 볼륨을 가져왔는지 여부와 PVC 및 PV가 관리되는지 여부를 나타내는 이중 목적의 주석이 추가됩니다. 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. `tridentctl import` 명령을 사용하여 볼륨이 포함된 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume)을 지정합니다. `-f` 인수는 PVC 파일의 경로를 지정하는 데 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.

예

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하십시오.

ONTAP NAS 및 ONTAP NAS FlexGroup

Trident는 `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

- Trident는 `ontap-nas-economy` 드라이버를 사용한 볼륨 가져오기를 지원하지 않습니다.
- `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버는 중복된 볼륨 이름을 허용하지 않습니다.

`ontap-nas` 드라이버를 사용하여 생성된 각 볼륨은 `ONTAP` 클러스터의 `FlexVol` 볼륨입니다. `ontap-nas` 드라이버를 사용하여 `FlexVol` 볼륨을 가져오는 과정도 동일합니다. `ONTAP` 클러스터에 이미 존재하는 `FlexVol` 볼륨은 `ontap-nas` PVC로 가져올 수 있습니다. 마찬가지로 `FlexGroup` 볼륨도 `ontap-nas-flexgroup` PVC로 가져올 수 있습니다.

tridentctl을 사용한 ONTAP NAS 예제

다음 예제는 `tridentctl`을 사용하여 관리형 볼륨과 관리되지 않는 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

다음 예시는 `managed_volume`라는 이름의 볼륨을 `ontap_nas`라는 이름의 백엔드에서 가져오는 방법을 보여줍니다:

```
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

`--no-manage` 인수를 사용할 때 Trident는 볼륨 이름을 변경하지 않습니다.

다음 예제는 unmanaged_volume`를 `ontap_nas 백엔드에서 가져옵니다:

```
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.

관리형 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 SAN

Trident는 `ontap-san`(iSCSI, NVMe/TCP 및 FC) 및 `ontap-san-economy`드라이버를 사용한 볼륨 가져오기를 지원합니다.

Trident는 단일 LUN을 포함하는 ONTAP SAN FlexVol 볼륨을 가져올 수 있습니다. 이는 각 PVC에 대해 FlexVol 볼륨을 생성하고 FlexVol 볼륨 내에 LUN을 생성하는 `ontap-san` 드라이버와 일치합니다. Trident는 FlexVol 볼륨을 가져와 PVC 정의와 연결합니다. Trident는 여러 LUN을 포함하는 `ontap-san-economy` 볼륨을 가져올 수 있습니다.

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

관리형 볼륨의 경우 Trident는 FlexVol 볼륨의 이름을 `pvc-<uuid>` 형식으로 변경하고 FlexVol 볼륨 내의 LUN 이름을 `lun0`로 변경합니다.

다음 예제는 `ontap-san-managed` FlexVol 볼륨을 가져옵니다 `ontap_san_default` 백엔드에 있는:

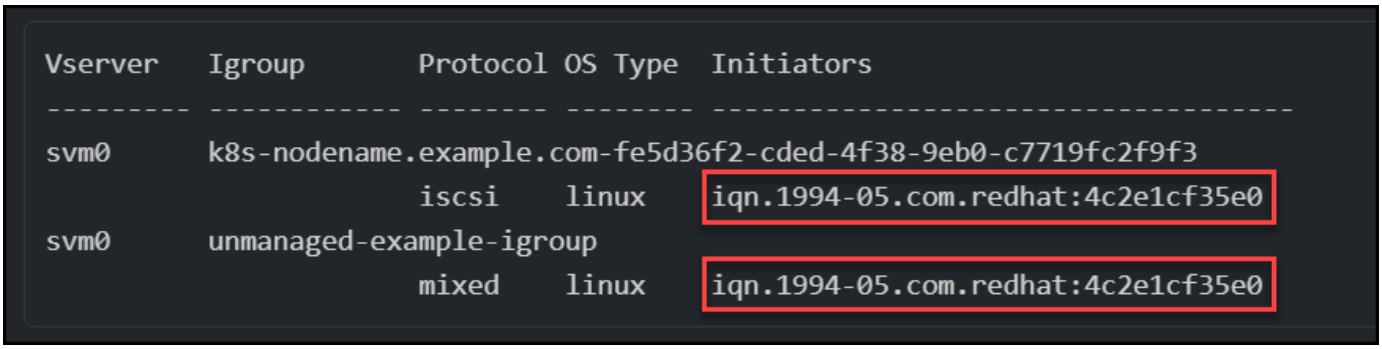
```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   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 예제는 `unmanaged_example_volume`를 `ontap_san` 백엔드에서 가져옵니다:

```
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 예시와 같이 Kubernetes 노드 IQN과 IQN을 공유하는 igroup에 매핑된 LUN이 있는 경우 다음 오류가 표시됩니다: `LUN already mapped to initiator(s) in this group`. 볼륨을 가져오려면 이니시에이터를 제거하거나 LUN 매핑을 해제해야 합니다.



ONTAP SAN-economy 예

다음 예제는 `ontap-san-economy` 백엔드에 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

관리형 볼륨을 가져오면 Trident가 FlexVol의 소유권을 가져오고 이름을 변경합니다. 동일한 FlexVol에서 여러 LUN을 가져올 때 이 이름 변경을 고려해야 합니다.

다음 예제는 FlexVol `toimport`에서 `lun1`를 가져와 관리되는 볼륨 `vol-managed-saneco`로 가져옵니다:

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

가져오기 `lun1` 후 Trident는 FlexVol의 이름을 변경합니다(예: `trident_lun_pool_xyz`). 동일한 FlexVol에서 추가 LUN을 가져오려면 새 FlexVol 이름을 사용하십시오.

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

`ontap-san-economy` 백엔드는 한 번에 하나의 LUN을 가져옵니다. 스크립트를 사용하여 여러 가져오기를 자동화할 수 있습니다.

관리되지 않는 볼륨

관리되지 않는 볼륨을 가져올 때 Trident는 FlexVol의 소유권을 가져오지 않습니다. 그러나 FlexVol과 LUN은 Trident 명명 규칙을 따라야 합니다.

FlexVol 명명 형식

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX`는 백엔드 구성의 `storagePrefix` 값입니다. 기본값은 `trident`입니다.
- `RANDOMSTRING`은(는) 선택한 문자열입니다.

LUN 명명 요구 사항

LUN의 이름은 `lun0`이어야 합니다.

예

`storagePrefix`이 `xyz`인 경우 LUN의 전체 경로는 다음과 같습니다.

```
trident_lun_pool_xyz_randomstring/lun0
```

요소

Trident는 `solidfire-san` 드라이버를 사용하여 NetApp Element 소프트웨어 및 NetApp HCI 볼륨 가져오기를 지원합니다.

Element 드라이버는 중복된 볼륨 이름을 지원합니다. 하지만 Trident는 중복된 볼륨 이름이 있는 경우 오류를 반환합니다. 해결 방법으로 볼륨을 복제하고 고유한 볼륨 이름을 지정한 다음 복제된 볼륨을 가져오십시오.

다음 예제는 백엔드 `element_default`에서 `element-managed` 볼륨을 가져옵니다.

```
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

Trident는 `azure-netapp-files` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

Azure NetApp Files 볼륨을 가져오려면 볼륨 경로를 사용하여 볼륨을 식별해야 합니다. 볼륨 경로는 볼륨 내보내기 경로에서 `:/` 뒤에 오는 부분입니다. 예를 들어 마운트 경로가 `10.0.0.2:/importvol1`인 경우 볼륨 경로는 `importvol1`입니다.

다음 예제는 백엔드에서 `azure-netapp-files` 볼륨을 가져오며, 볼륨 경로는 `azurenetappfiles_40517`입니다 `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

Trident는 `google-cloud-netapp-volumes` 드라이버를 사용하여 볼륨 가져오기를 지원합니다.

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 예제는 동일한 영역에 두 개의 볼륨이 있는 경우 `google-cloud-netapp-volumes` 볼륨을 가져옵니다.

```
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 지정

Trident를 사용하면 생성하는 볼륨에 의미 있는 이름과 레이블을 지정할 수 있습니다. 이를 통해 볼륨을 쉽게 식별하고 해당 Kubernetes 리소스(PVC)에 매핑할 수 있습니다. 또한 백엔드 수준에서 사용자 지정 볼륨 이름과 사용자 지정 레이블을 생성하기 위한 템플릿을 정의할 수 있으며, 생성, 가져오기 또는 복제하는 모든 볼륨은 이러한 템플릿을 따릅니다.

시작하기 전에

사용자 지정 가능한 볼륨 이름 및 레이블 지원:

- 볼륨 생성, 가져오기 및 클론 작업.
- `ontap-nas-economy` 드라이버의 경우 Qtree 볼륨의 이름만 이름 템플릿을 준수합니다.
- `ontap-san-economy` 드라이버의 경우 LUN 이름만 이름 템플릿을 준수합니다.

제한 사항

- 사용자 지정 볼륨 이름은 ONTAP 온프레미스 드라이버에서만 호환됩니다.
- 사용자 지정 레이블은 `ontap-san`, `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버에서만 지원됩니다.
- 사용자 지정 볼륨 이름은 기존 볼륨에 적용되지 않습니다.

사용자 지정 가능한 볼륨 이름의 주요 동작

- 이름 템플릿의 구문 오류로 인해 오류가 발생하면 백엔드 생성이 실패합니다. 하지만 템플릿 적용이 실패할 경우 볼륨 이름은 기존 명명 규칙에 따라 지정됩니다.
- 백엔드 구성의 이름 템플릿을 사용하여 볼륨 이름을 지정할 때는 스토리지 접두사를 적용할 수 없습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블이 포함된 백엔드 구성 예

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

루트 레벨 예

```
{
  "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 수준 예

```
{
  "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 예

예 **1**:

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

예 **2**:

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

고려해야 할 사항

1. 볼륨 가져오기의 경우, 기존 볼륨에 특정 형식의 레이블이 있는 경우에만 레이블이 업데이트됩니다. 예를 들면 다음과 같습니다: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. 관리형 볼륨 가져오기의 경우 볼륨 이름은 백엔드 정의의 루트 수준에 정의된 이름 템플릿을 따릅니다.
3. Trident는 스토리지 접두사와 함께 슬라이스 연산자를 사용하는 것을 지원하지 않습니다.
4. 템플릿을 사용했을 때 고유한 볼륨 이름이 생성되지 않으면 Trident가 임의의 문자를 추가하여 고유한 볼륨 이름을 생성합니다.
5. NAS 이코노미 볼륨의 사용자 지정 이름이 64자를 초과하면 Trident는 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

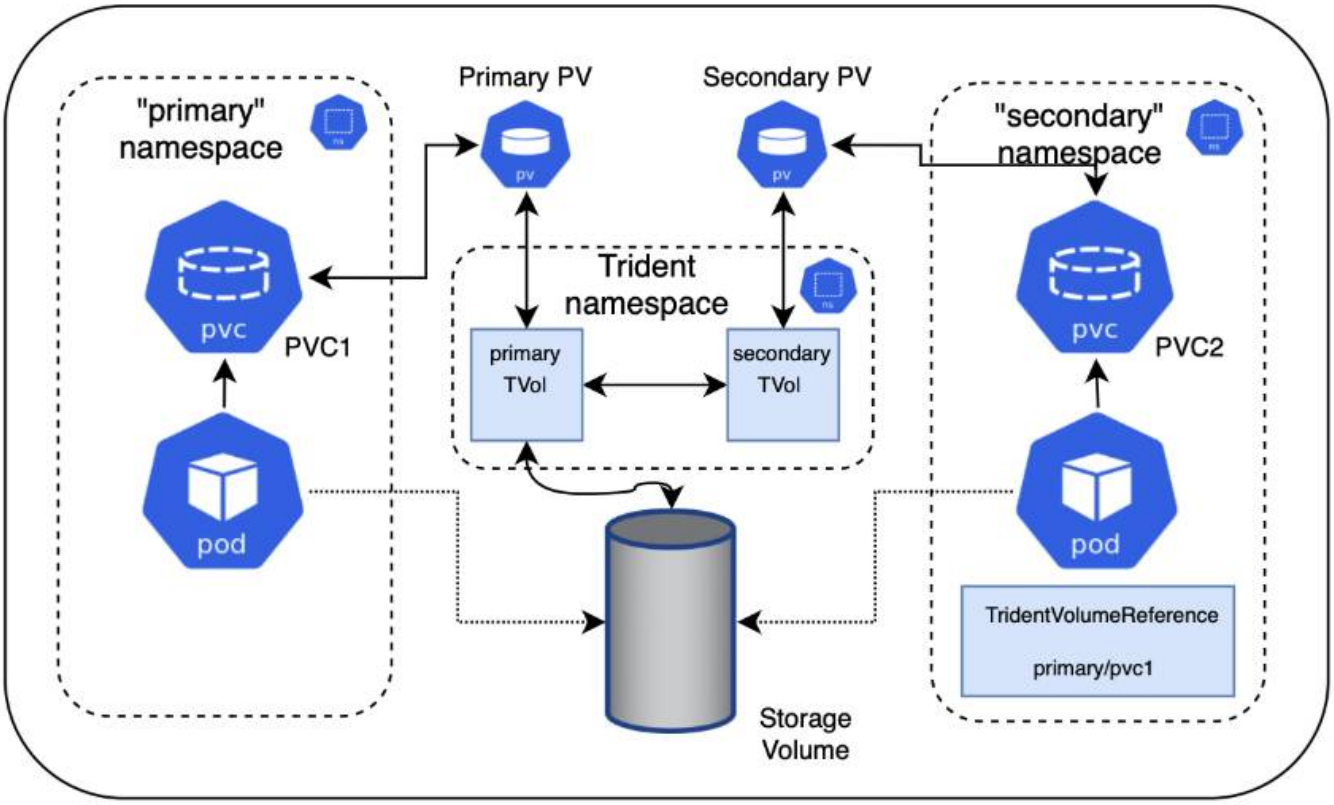
Trident를 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

기능

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

- 보안을 보장하기 위한 여러 수준의 액세스 제어
- 모든 Trident NFS 볼륨 드라이버와 호환됩니다.
- tridentctl 또는 기타 비네이티브 Kubernetes 기능에 의존하지 않음

이 다이어그램은 두 Kubernetes 네임스페이스에서 NFS 볼륨 공유를 보여줍니다.



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1 소스 **PVC**를 구성하여 볼륨을 공유합니다

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

2 대상 네임스페이스에 **CR**을 생성할 수 있는 권한을 부여하십시오

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 생성합니다

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

소스 및 대상 네임스페이스를 구성합니다

보안을 보장하기 위해 네임스페이스 간 공유는 소스 네임스페이스 소유자, 클러스터 관리자 및 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계에서 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: 소스 네임스페이스에서 PVC (`pvc1`)를 생성하여 대상 네임스페이스와 공유할 수 있는 권한을 부여합니다 (`namespace2`). 이때 `shareToNamespace` 주석을 사용합니다.

```
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
```

Trident는 PV와 해당 백엔드 NFS 스토리지 볼륨을 생성합니다.

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

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 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: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, 소스 PVC를

지정하기 위해 `shareFromPVC` 주석을 사용합니다.

```
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

Trident는 대상 PVC의 `shareFromPVC` 어노테이션을 읽고 소스 PV를 가리키고 소스 PV 스토리지 리소스를 공유하는 자체 스토리지 리소스가 없는 하위 볼륨으로 대상 PV를 생성합니다. 대상 PVC와 PV는 정상적으로 바인딩된 것처럼 보입니다.

공유 볼륨 삭제

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

`tridentctl get`를 사용하여 하위 볼륨을 쿼리합니다

이 [`tridentctl` 유틸리티를 사용하면 `get` 명령을 실행하여 하위 볼륨을 가져올 수 있습니다. 자세한 내용은 `tridentctl` 명령 및 옵션을 참조하십시오.

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

플래그:

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

제한 사항

- Trident는 대상 네임스페이스가 공유 볼륨에 쓰기 작업을 하는 것을 막을 수 없습니다. 공유 볼륨 데이터가 덮어쓰이는 것을 방지하려면 파일 잠금 또는 다른 프로세스를 사용해야 합니다.
- `shareToNamespace` 또는 `shareFromNamespace` 어노테이션을 제거하거나 `TridentVolumeReference` CR을 삭제해도 소스 PVC에 대한 액세스 권한을 취소할 수 없습니다. 액세스 권한을 취소하려면 하위 PVC를 삭제해야 합니다.
- 하위 볼륨에서는 스냅샷, 클론 및 미러링이 불가능합니다.

자세한 내용은

교차 네임스페이스 볼륨 액세스에 대한 자세한 내용은 다음을 참조하십시오.

- "NetAppTV"에서 데모를 시청하십시오.

네임스페이스 간 볼륨 복제

Trident를 사용하면 동일한 Kubernetes 클러스터 내의 다른 네임스페이스에 있는 기존 볼륨 또는 볼륨 스냅샷을 사용하여 새 볼륨을 생성할 수 있습니다.

필수 구성 요소

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

> ⓘ 네임스페이스 간 복제는 `ontap-san` 및 `ontap-nas` 스토리지 드라이버에 한해서만 지원됩니다. 읽기 전용 복제는 지원되지 않습니다.

빠른 시작

몇 단계만 거치면 볼륨 클로닝을 설정할 수 있습니다.

1. 소스 **PVC**를 구성하여 볼륨 복제

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

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

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 클론 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 소스 네임스페이스의 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스를 구성합니다

보안을 보장하기 위해 네임스페이스 간 볼륨 복제에는 소스 네임스페이스 소유자, 클러스터 관리자 및 대상 네임스페이스 소유자의 협업과 조치가 필요합니다. 각 단계에서 사용자 역할이 지정됩니다.

단계

1. 소스 네임스페이스 소유자: 소스 네임스페이스(`namespace1`)에서 PVC(`pvc1`)를 생성하고, `cloneToNamespace`주석을 사용하여 대상 네임스페이스(`namespace2`)와 공유할 수 있는 권한을 부여합니다.

```
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
```

Trident가 PV와 해당 백엔드 스토리지 볼륨을 생성합니다.

> ◦ 쉼표로 구분된 목록을 사용하여 여러 네임스페이스에서 PVC를 공유할 수 있습니다. 예를 들어 `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
> ◦ `*`를 사용하여 모든 네임스페이스에 공유할 수 있습니다. 예를 들어 `trident.netapp.io/cloneToNamespace: *`
> ◦ 언제든지 PVC를 업데이트하여 `cloneToNamespace` 주석을 포함시킬 수 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오(`namespace2`.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 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: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, `cloneFromPVC` 또는 `cloneFromSnapshot` 및 `cloneFromNamespace` 주석을 사용하여 소스 PVC를 지정합니다.

```
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 클론은 지원되지 않습니다.

SnapMirror를 사용하여 볼륨 복제

Trident는 재해 복구를 위한 데이터 복제를 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간의 미러 관계를 지원합니다. Trident Mirror Relationship(TMR)이라는 네임스페이스 Custom Resource Definition(CRD)을 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨(PVC) 간의 미러 관계 생성
- 볼륨 간의 미러 관계를 제거합니다
- 미러 관계를 중단합니다
- 재해 조건(페일오버) 중에 보조 볼륨을 승격합니다

- 클러스터 간 애플리케이션의 무손실 전환 수행(계획된 페일오버 또는 마이그레이션 중)

복제 사전 요구 사항

시작하기 전에 다음 사전 요구 사항이 충족되었는지 확인하십시오.

ONTAP 클러스터

- **Trident**: ONTAP를 백엔드로 사용하는 소스 및 타겟 Kubernetes 클러스터 모두에 Trident 버전 22.10 이상이 있어야 합니다.
- 라이선스: 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스는 소스 및 타겟 ONTAP 클러스터 모두에서 활성화되어야 합니다. 자세한 내용은 "SnapMirror ONTAP 라이센싱 개요"을 참조하십시오.

  ONTAP 9.10.1부터 모든 라이센스는 여러 기능을 활성화하는 단일 파일인 NetApp 라이센스 파일(NLF)로 제공됩니다. 자세한 내용은 "ONTAP One에 포함된 라이센스"을 참조하십시오.

  SnapMirror 비동기 보호 기능만 지원됩니다.

피어링

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

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인하십시오.

- **Trident** 및 **SVM**: 피어링된 원격 SVM은 타겟 클러스터의 Trident에서 사용할 수 있어야 합니다.

지원되는 드라이버

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

SnapMirror를 사용한 볼륨 복제는 ASA r2 시스템에서 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 "ASA r2 스토리지 시스템에 대해 알아보세요"를 참조하십시오.

미러링된 **PVC** 생성

다음 단계를 따르고 CRD 예제를 사용하여 운영 볼륨과 2차 볼륨 간의 미러 관계를 생성하십시오.

단계

1. 기본 Kubernetes 클러스터에서 다음 단계를 수행하십시오.
   a. `trident.netapp.io/replication: true` 매개 변수를 사용하여 StorageClass 객체를 생성합니다.

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. 로컬 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

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

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

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻으십시오.

```
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 수행하십시오.
   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. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

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"
```

Trident는 구성된 관계 정책 이름(또는 ONTAP의 기본값)으로 SnapMirror 관계를 생성하고 초기화합니다.

c. 이전에 생성한 StorageClass를 사용하여 보조(SnapMirror 타겟) 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
```

Trident는 TridentMirrorRelationship CRD를 확인하고, 관계가 존재하지 않으면 볼륨 생성을 실패합니다. 관계가 존재하는 경우, Trident는 새 FlexVol 볼륨이 MirrorRelationship에 정의된 원격 SVM과 피어링된 SVM에 배치되도록 합니다.

볼륨 복제 상태

Trident 미러 관계(TMR)는 PVC 간의 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 원하는 상태를 Trident에 알려주는 상태가 있습니다. 대상 TMR은 다음과 같은 상태를 가집니다.

- **Established**: 로컬 PVC는 미러 관계의 타겟 볼륨이며, 이는 새로운 관계입니다.
- 승격됨: 로컬 PVC는 ReadWrite이며 마운트 가능하고, 현재 미러 관계가 적용되지 않습니다.
- 재설정됨: 로컬 PVC는 미러 관계의 타겟 볼륨이며 이전에도 해당 미러 관계에 있었습니다.

◦ 타겟 볼륨이 소스 볼륨과 관계를 맺은 적이 있는 경우 타겟 볼륨 콘텐츠를 덮어쓰므로 재설정된 상태를 사용해야 합니다.

◦ 볼륨이 이전에 소스와 관계를 맺지 않았던 경우 재설정된 상태가 실패합니다.

예기치 않은 페일오버 시 보조 PVC 승격

보조 Kubernetes 클러스터에서 다음 단계를 수행하십시오.

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

계획된 페일오버 중에 보조 PVC를 승격합니다

계획된 페일오버(마이그레이션) 중에 다음 단계를 수행하여 보조 PVC를 승격시키십시오.

단계

1. 기본 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 기본 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 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 promoted로 승격되었는지 확인합니다.

페일오버 후 미러 관계 복원

미러 관계를 복원하기 전에 새 운영 환경으로 지정할 측을 선택합니다.

단계

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

추가 작업

Trident는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

기본 **PVC**를 새 보조 **PVC**로 복제합니다

기본 PVC와 보조 PVC가 이미 있는지 확인하십시오.

단계

1. 설정된 보조(대상) 클러스터에서 PersistentVolumeClaim 및 TridentMirrorRelationship CRD를 삭제합니다.
2. 기본(소스) 클러스터에서 TridentMirrorRelationship CRD를 삭제합니다.
3. 설정하려는 새로운 보조(대상) PVC에 대해 기본(소스) 클러스터에 새 TridentMirrorRelationship CRD를 생성합니다.

미러링된 운영 또는 보조 **PVC**의 크기 조정

PVC는 정상적으로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과하면 ONTAP은 대상 FlexVol을 자동으로 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행하십시오.

- 보조 PVC에서 MirrorRelationship을 삭제합니다. 이렇게 하면 복제 관계가 끊어집니다.
- 또는 spec.state 필드를 _promoted_로 업데이트하십시오.

PVC 삭제**(**이전에 미러링됨**)**

Trident는 복제된 PVC를 확인하고 볼륨 삭제를 시도하기 전에 복제 관계를 해제합니다.

TMR 삭제

미러링 관계의 한쪽에서 TMR을 삭제하면 Trident가 삭제를 완료하기 전에 나머지 TMR이 *promoted* 상태로 전환됩니다. 삭제 대상으로 선택된 TMR이 이미 *promoted* 상태인 경우 기존 미러 관계가 없으므로 해당 TMR이 제거되고 Trident는 로컬 PVC를 *ReadWrite_*로 승격시킵니다. 이 삭제는 *ONTAP*의 로컬 볼륨에 대한 *SnapMirror* 메타데이터를 해제합니다. 향후 이 볼륨을 미러 관계에 사용하려면 새 미러 관계를 생성할 때 *_established* 볼륨 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP이 온라인 상태일 때 미러 관계를 업데이트합니다

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

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

Trident가 ONTAP 클러스터에 직접 연결되지 않은 상태에서 CRD를 사용하여 SnapMirror 업데이트를 수행할 수 있습니다. 다음 TridentActionMirrorUpdate 예제 형식을 참조하십시오.

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 나타냅니다. *Succeeded*, *In Progress*, 또는 *Failed* 값을 가질 수 있습니다.

CSI 토폴로지 사용

Trident은 "CSI 토폴로지 기능"을 사용하여 Kubernetes 클러스터에 있는 노드에 선택적으로 볼륨을 생성하고 연결할 수 있습니다.

개요

CSI 토폴로지 기능을 사용하면 리전 및 가용 영역을 기반으로 볼륨 액세스를 노드 하위 집합으로 제한할 수 있습니다. 오늘날 클라우드 제공업체는 Kubernetes 관리자가 영역 기반 노드를 생성할 수 있도록 지원합니다. 노드는 리전 내의 서로 다른 가용 영역에 위치하거나 여러 리전에 걸쳐 위치할 수 있습니다. 다중 영역 아키텍처에서 워크로드에 대한 볼륨 프로비저닝을 용이하게 하기 위해 Trident는 CSI 토폴로지를 사용합니다.

CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- `VolumeBindingMode`을 `Immediate`로 설정하면 Trident는 토폴로지를 고려하지 않고 볼륨을 생성합니다. 볼륨 바인딩 및 동적 프로비저닝은 PVC 생성 시 처리됩니다. 이는 기본 `VolumeBindingMode`이며 토폴로지 제약 조건을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청하는 Pod의 스케줄링 요구 사항에 관계없이 생성됩니다.
- `VolumeBindingMode`을 `WaitForFirstConsumer`로 설정하면 PVC에 대한 영구 볼륨의 생성 및 바인딩이 해당 PVC를 사용하는 파드가 스케줄링되고 생성될 때까지 지연됩니다. 이렇게 하면 토폴로지 요구 사항에 따라 적용되는 스케줄링 제약 조건을 충족하도록 볼륨이 생성됩니다.

`WaitForFirstConsumer` 바인딩 모드는 토폴로지 레이블을 필요로 하지 않습니다. CSI 토폴로지 기능과 별개로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- "지원되는 Kubernetes 버전"를 실행하는 Kubernetes 클러스터

```
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 도입하는 레이블이 있어야 합니다(`topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`). 이러한 레이블은 Trident가 토폴로지를 인식하도록 하려면 Trident를 설치하기 전에 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 생성

Trident 스토리지 백엔드는 가용성 영역을 기반으로 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드는 지원되는 영역 및 리전 목록을 나타내는 선택적 `supportedTopologies` 블록을 포함할 수 있습니다. 이러한 백엔드를 사용하는 StorageClasses의 경우, 지원되는 리전/영역에 스케줄링된 애플리케이션에서 요청한 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`은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 이러한 리전 및 영역은 StorageClass에 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 StorageClasses의 경우 Trident는 해당 백엔드에 볼륨을 생성합니다.

`supportedTopologies`를 스토리지 풀별로 정의할 수도 있습니다. 다음 예를 참조하십시오.

```
---
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 예에서 `region` 및 `zone` 레이블은 스토리지 풀의 위치를 나타냅니다. `topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`는 스토리지 풀을 사용할 수 있는 위치를 지정합니다.

2단계: 토폴로지를 인식하는 StorageClasses를 정의합니다

클러스터의 노드에 제공되는 토폴로지 레이블을 기반으로, StorageClasses를 정의하여 토폴로지 정보를 포함할 수 있습니다. 이를 통해 PVC 요청에 대한 후보로 사용되는 스토리지 풀과 Trident에서 프로비저닝한 볼륨을 사용할 수 있는 노드의 하위 집합이 결정됩니다.

다음 예를 참조하십시오.

```
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`는 `WaitForFirstConsumer`로 설정되어 있습니다. 이 `StorageClass`로 요청된 `PVC`는 `Pod`에서 참조될 때까지 처리되지 않습니다. 그리고 `allowedTopologies`는 사용할 영역과 지역을 제공합니다. `netapp-san-us-east1` StorageClass는 위에 정의된 `san-backend-us-east1` 백엔드에 PVC를 생성합니다.

3단계: PVC 생성 및 사용

StorageClass를 생성하고 백엔드에 매핑했으면 이제 PVC를 생성할 수 있습니다.

아래 예시를 참조하세요 `spec`:

```
---
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 나타납니다.

```
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
```

Trident가 볼륨을 생성하고 PVC에 바인딩하려면 포드에서 PVC를 사용하십시오. 다음 예를 참조하십시오.

```
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 Kubernetes에게 `us-east1` 지역에 있는 노드에 Pod를 예약하도록 지시하며, `us-east1-a` 또는 `us-east1-b` 영역에 있는 모든 노드 중에서 선택합니다.

다음 출력을 참조하십시오.

```
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 다음을 포함하도록 합니다 `supportedTopologies`

기존 백엔드를 업데이트하여 `supportedTopologies`목록을 포함할 수 있습니다 `tridentctl backend update`. 이는 이미 프로비저닝된 볼륨에는 영향을 미치지 않으며 이후 PVC에만 사용됩니다.

자세한 정보 찾기

- "컨테이너의 리소스 관리"
- "nodeSelector"
- "친화성 및 반친화성"
- "Taints 및 Tolerations"

스냅샷 작업

Kubernetes 볼륨 스냅샷의 영구 볼륨(PVs)은 볼륨의 시점 복사본을 활성화합니다. Trident를 사용하여 생성된 볼륨의 스냅샷을 생성하고, Trident 외부에서 생성된 스냅샷을 가져오고, 기존 스냅샷에서 새 볼륨을 생성하고, 스냅샷에서 볼륨 데이터를 복구할 수 있습니다.

개요

볼륨 스냅샷은 `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` 및 `google-cloud-netapp-volumes` 드라이버에서 지원됩니다.

시작하기 전에

스냅샷으로 작업하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. 이는 Kubernetes 오케스트레이터의 책임입니다(예: Kubeadm, GKE, OpenShift).

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 볼륨 스냅샷 컨트롤러를 배포합니다을 참조하십시오.

> GKE 환경에서 온디맨드 볼륨 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

볼륨 스냅샷을 생성합니다

단계

1. `VolumeSnapshotClass`을(를) 생성합니다. 자세한 내용은 "VolumeSnapshotClass"을(를) 참조하십시오.
   - `driver`는 Trident CSI 드라이버를 가리킵니다.
   - `deletionPolicy`은 `Delete 또는 Retain`일 수 있습니다. `Retain`로 설정하면 `VolumeSnapshot 객체가 삭제되더라도 스토리지 클러스터의 기본 물리적 스냅샷이 유지됩니다.

     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 생성합니다.

   예

   - 이 예제는 기존 PVC의 스냅샷을 생성합니다.

     ```
     cat snap.yaml
     ```

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

   - 이 예제는 pvc1`라는 이름의 PVC에 대한 볼륨 스냅샷 객체를 생성하고 스냅샷 이름을 `pvc1-snap`로 설정합니다. VolumeSnapshot은 PVC와 유사하며 실제 스냅샷을 나타내는 `VolumeSnapshotContent 객체와 연결됩니다.

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

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

- `VolumeSnapshotContent` 객체를 `pvc1-snap` VolumeSnapshot에 대해 설명하여 식별할 수 있습니다. `Snapshot Content Name`는 이 스냅샷을 제공하는 `VolumeSnapshotContent` 객체를 식별합니다. `Ready To Use` 매개변수는 스냅샷을 사용하여 새 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**를 생성합니다

`dataSource`을 사용하여 `VolumeSnapshot`이라는 이름의 `<pvc-name>`를 데이터 소스로 사용하는 `PVC`를 생성할 수 있습니다. `PVC`가 생성되면 `Pod`에 연결하여 다른 `PVC`와 마찬가지로 사용할 수 있습니다.

PVC는 소스 볼륨과 동일한 백엔드에서 생성됩니다. 다음을 참조하십시오. "KB: Trident PVC 스냅샷에서 PVC를 생성하는 작업은 다른 백엔드에서 수행할 수 없습니다".

다음 예제는 `pvc1-snap`를 데이터 소스로 사용하여 PVC를 생성합니다.

```
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

Trident는 "Kubernetes 사전 프로비저닝된 스냅샷 프로세스"을 지원하여 클러스터 관리자가 `VolumeSnapshotContent` 객체를 생성하고 Trident 외부에서 생성된 스냅샷을 가져올 수 있도록 합니다.

시작하기 전에

Trident가 스냅샷의 상위 볼륨을 생성하거나 가져와야 합니다.

단계

1. 클러스터 관리자: 백엔드 스냅샷을 참조하는 `VolumeSnapshotContent` 객체를 생성합니다. 이렇게 하면 Trident에서 스냅샷 워크플로우가 시작됩니다.
   - 백엔드 스냅샷의 이름을 `annotations`에서 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`로 지정하세요.
   - `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`를 `snapshotHandle`에 지정하십시오. 이것이 `ListSnapshots` 호출에서 외부 스냅샷터가 Trident에 제공하는 유일한 정보입니다.

     `<volumeSnapshotContentName>`는 CR 명명 제약 조건으로 인해 백엔드 스냅샷 이름과 항상 일치하는 것은 아닙니다.

     예

     다음 예제는 백엔드 스냅샷 `snap-01`을 참조하는 `VolumeSnapshotContent` 오브젝트를 생성합니다.

```
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: `VolumeSnapshot` 객체를 참조하는 CR을 생성합니다. `VolumeSnapshotContent` 이것은 지정된 네임스페이스에서 `VolumeSnapshot` 사용 권한을 요청합니다.

   예

   다음 예제는 `VolumeSnapshot`라는 이름의 CR을 생성하며, `import-snap`를 참조하고, `VolumeSnapshotContent`라는 이름의 `import-snap-content`를 참조합니다.

   ```
   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(별도 조치 필요 없음): 외부 스냅샷 생성기가 새로 생성된 `VolumeSnapshotContent`을 인식하고 `ListSnapshots` 호출을 실행합니다. Trident가 `TridentSnapshot`을 생성합니다.

   - 외부 snapshotter는 `VolumeSnapshotContent`를 `readyToUse`로, `VolumeSnapshot`를 `true`로 설정합니다.
   - Trident가 반환됩니다 `readyToUse=true`.

4. 모든 사용자: 새 `PersistentVolumeClaim`를 참조하기 위해 `VolumeSnapshot`를 생성합니다. 여기서 `spec.dataSource` (또는 `spec.dataSourceRef`) 이름은 `VolumeSnapshot` 이름입니다.

   예

다음 예제는 `VolumeSnapshot`라는 이름의 `import-snap`을(를) 참조하는 PVC를 생성합니다.

```
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 디렉터리는 기본적으로 숨겨져 있어 `ontap-nas` 및 `ontap-nas-economy` 드라이버를 사용하여 프로비저닝된 볼륨의 최대 호환성을 용이하게 합니다. 스냅샷에서 직접 데이터를 복구하려면 `.snapshot` 디렉터리를 활성화하십시오.

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 대한 변경 사항은 손실됩니다.

스냅샷에서 볼륨 제자리 복원

Trident는 `TridentActionSnapshotRestore` (TASR) CR을 사용하여 스냅샷에서 신속하게 제자리 볼륨 복원을 제공합니다. 이 CR은 명령형 Kubernetes 작업으로 작동하며 작업 완료 후에는 지속되지 않습니다.

Trident는 `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` 및 `solidfire-san` 드라이버에서 스냅샷 복원을 지원합니다.

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인합니다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예제에서는 PVC `pvc1` 및 볼륨 스냅샷 `pvc1-snapshot`용 CR을 생성합니다.

   TASR CR은 PVC 및 VS가 존재하는 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 적용하세요. 이 예에서는 스냅샷에서 복원합니다 `pvc1`.

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

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

결과

Trident 스냅샷에서 데이터를 복원합니다. 스냅샷 복원 상태는 다음과 같이 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 Trident는 실패 시 자동으로 작업을 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 액세스 권한이 없는 Kubernetes 사용자는 관리자로부터 애플리케이션 네임스페이스에 TASR CR을 생성할 수 있는 권한을 부여받아야 할 수도 있습니다.

연결된 스냅샷이 있는 PV 삭제

스냅샷이 연결된 영구 볼륨을 삭제하면 해당 Trident 볼륨의 상태가 "삭제 중"으로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨 스냅샷을 제거하십시오.

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 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 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "볼륨 스냅샷"
- "VolumeSnapshotClass"

볼륨 그룹 스냅샷 작업

퍼시스턴트 볼륨(PV)의 Kubernetes 볼륨 그룹 스냅샷 NetApp Trident는 여러 볼륨의 스냅샷(볼륨 스냅샷 그룹)을 생성하는 기능을 제공합니다. 이 볼륨 그룹 스냅샷은 동일한 시점에 생성된 여러 볼륨의 복사본을 나타냅니다.

VolumeGroupSnapshot은 베타 API를 사용하는 Kubernetes의 베타 기능입니다. VolumeGroupSnapshot에 필요한 최소 버전은 Kubernetes 1.32입니다.

볼륨 그룹 스냅샷 만들기

볼륨 그룹 스냅샷은 다음 스토리지 드라이버에서 지원됩니다.

- `ontap-san` driver - iSCSI 및 FC 프로토콜에만 해당되며 NVMe/TCP 프로토콜에는 해당되지 않습니다.
- `ontap-san-economy` - iSCSI 프로토콜에만 해당합니다.
- `ontap-nas`

볼륨 그룹 스냅샷은 NetApp ASA r2 또는 AFX 스토리지 시스템에서는 지원되지 않습니다.

시작하기 전에

- Kubernetes 버전이 K8s 1.32 이상인지 확인합니다.
- 스냅샷으로 작업하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. 이는 Kubernetes 오케스트레이터의 책임입니다(예: Kubeadm, GKE, OpenShift).

  Kubernetes 배포에 외부 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 볼륨 스냅샷 컨트롤러를 배포합니다을 참조하십시오.

  GKE 환경에서 온디맨드 볼륨 그룹 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

- 스냅샷 컨트롤러 YAML에서 `CSIVolumeGroupSnapshot` 기능 게이트를 'true'로 설정하여 볼륨 그룹 스냅샷이 활성화되도록 합니다.
- 볼륨 그룹 스냅샷을 생성하기 전에 필요한 볼륨 그룹 스냅샷 클래스를 생성합니다.
- 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하여 VolumeGroupSnapshot을 생성할 수 있도록 합니다.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하십시오. 자세한 내용은 "VolumeGroupSnapshotClass"을 참조하십시오.

```
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 스토리지 클래스를 사용하여 필수 레이블이 포함된 PVC를 생성하거나 기존 PVC에 이러한 레이블을 추가합니다.

  다음 예는 `pvc1-group-snap`을 데이터 소스로 사용하고 `consistentGroupSnapshot: groupA`을 레이블로 사용하여 PVC를 생성합니다. 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
```

- PVC에 지정된 것과 동일한 레이블(`consistentGroupSnapshot: groupA`)을 사용하여 VolumeGroupSnapshot을 만듭니다.

  이 예에서는 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 하나의 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 대한 변경 사항은 손실됩니다.

스냅샷에서 볼륨 제자리 복원

Trident는 `TridentActionSnapshotRestore` (TASR) CR을 사용하여 스냅샷에서 신속하게 제자리 볼륨 복원을 제공합니다. 이 CR은 명령형 Kubernetes 작업으로 작동하며 작업 완료 후에는 지속되지 않습니다.

자세한 내용은 "스냅샷에서 볼륨 제자리 복원"을 참조하십시오.

연결된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- VolumeGroupSnapshots는 그룹 전체로 삭제할 수 있지만, 그룹 내의 개별 스냅샷은 삭제할 수 없습니다.
- PersistentVolumes가 해당 PersistentVolume에 대한 스냅샷이 존재하는 동안 삭제되면 Trident는 해당 볼륨을 "삭제 중" 상태로 이동합니다. 이는 볼륨을 안전하게 제거하기 전에 스냅샷을 먼저 제거해야 하기 때문입니다.
- 그룹 스냅샷을 사용하여 클론을 생성한 후 해당 그룹을 삭제하려는 경우, 클론 분할 작업이 시작되며 분할이 완료될 때까지 그룹을 삭제할 수 없습니다.

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포판에 스냅샷 컨트롤러와 CRD가 포함되어 있지 않은 경우 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 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "VolumeGroupSnapshotClass"
- "볼륨 스냅샷"

저작권 정보

상표 정보