



볼륨을 프로비저닝하고 관리합니다

Trident

NetApp
March 12, 2026

목차

볼륨을 프로비저닝하고 관리합니다	1
볼륨을 프로비저닝합니다	1
개요	1
PVC를 작성합니다	1
볼륨 확장	4
iSCSI 볼륨을 확장합니다	4
FC 볼륨을 확장합니다	8
NFS 볼륨을 확장합니다	12
RWX NVMe 하위 시스템 제한 사항 이해	15
64개 노드 제한 이해	15
NVMe 하위 시스템 모델 이해	15
오류 증상 파악	16
하위 시스템 제한 오류 해결	16
Trident를 업그레이드하여 슈퍼 서브시스템 모델을 적용합니다	16
컨트롤러 확장성	17
핵심 개념 및 정의	17
컨트롤러 확장성 지원	17
컨트롤러 확장성 활성화	17
동시성 동작	20
알려진 제한 사항 및 고려 사항	20
주의사항 및 제한사항	20
권장 사항	21
자동 볼륨 확장	21
요구 사항	21
제한 사항	21
자동 확장 정책이 적용된 볼륨 프로비저닝	22
자동 증가 정책 생성	22
자동 증가 정책 생성	23
정책 상태	23
정책을 StorageClass에 연결합니다	24
정책 우선 순위	25
구성 예	25
볼륨 가져오기	28
개요 및 고려 사항	28
볼륨을 가져옵니다	29
예	31
볼륨 이름 및 레이블을 사용자 지정합니다	38
시작하기 전에	38
제한 사항	38

사용자 지정 가능한 볼륨 이름의 주요 동작	39
이름 템플릿 및 레이블이 있는 백엔드 구성 예	39
템플릿 예제를 명명합니다	40
고려해야 할 사항	41
네임스페이스 전체에서 NFS 볼륨을 공유합니다	41
피처	41
빠른 시작	42
소스 및 대상 네임스페이스를 구성합니다	43
공유 볼륨을 삭제합니다	44
사용 <code>tridentctl get</code> 하위 볼륨을 쿼리합니다	44
제한 사항	45
를 참조하십시오	45
네임스페이스 전체에 걸친 클론 볼륨	45
필수 구성 요소	45
빠른 시작	45
소스 및 대상 네임스페이스를 구성합니다	46
제한 사항	47
SnapMirror를 사용하여 볼륨을 복제합니다	48
복제 사전 요구 사항	48
대칭 복사된 PVC를 작성합니다	49
볼륨 복제 상태입니다	52
비계획 파일오버 중에 보조 PVC를 승격합니다	52
계획된 파일오버 중에 보조 PVC를 승격합니다	52
파일오버 후 미리 관계를 복구합니다	53
추가 작업	53
ONTAP가 온라인 상태일 때 미리 관계를 업데이트합니다	54
ONTAP이 오프라인일 때 미리 관계를 업데이트합니다	54
CSI 토폴로지를 사용합니다	54
개요	54
1단계: 토폴로지 인식 백엔드 생성	56
2단계: 토폴로지를 인식하는 StorageClasses를 정의합니다	58
3단계: PVC 생성 및 사용	59
포함할 백엔드를 업데이트합니다 <code>supportedTopologies</code>	62
자세한 내용을 확인하십시오	62
스냅샷 작업	62
개요	62
볼륨 스냅샷을 생성합니다	63
볼륨 스냅샷에서 PVC를 생성합니다	64
볼륨 스냅샷을 가져옵니다	65
스냅샷을 사용하여 볼륨 데이터를 복구합니다	67
스냅샷에서 전체 볼륨 복원	67

연결된 스냅샷이 있는 PV를 삭제합니다	69
볼륨 스냅샷 컨트롤러를 배포합니다	69
관련 링크	70
볼륨 그룹 스냅샷 작업	70
볼륨 그룹 스냅샷 생성	71
그룹 스냅샷을 사용하여 볼륨 데이터 복구	72
스냅샷에서 전체 볼륨 복원	73
연관된 그룹 스냅샷이 있는 PV 삭제	73
볼륨 스냅샷 컨트롤러를 배포합니다	73
관련 링크	74

볼륨을 프로비저닝하고 관리합니다

볼륨을 프로비저닝합니다

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

개요

A "*PersistentVolumeClaim*"(PVC)는 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

PVC는 특정 크기 또는 액세스 모드의 저장을 요청하도록 구성할 수 있습니다. 클러스터 관리자는 연결된 StorageClass를 사용하여 PersistentVolume 크기 및 액세스 모드(예: 성능 또는 서비스 수준)를 제어할 수 있습니다.

PVC를 생성한 후 포드에 볼륨을 마운트할 수 있습니다.

PVC를 작성합니다

단계

1. PVC를 작성합니다.

```
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. 볼륨을 Pod에 마운트합니다.

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



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

2. 볼륨이 에 마운트되어 있는지 확인합니다 `/my/mount/path`.

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

3. 이제 Pod를 삭제할 수 있습니다. Pod 응용 프로그램은 더 이상 존재하지 않지만 볼륨은 유지됩니다.

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

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

이러한 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWO 액세스 PVC

이 예에서는 이름이 인 StorageClass와 연결된 RWO 액세스 권한이 있는 기본 PVC를 보여 줍니다 basic-csi.

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

NVMe/TCP가 있는 PVC

이 예에서는 이름이 인 StorageClass와 연결된 RWO 액세스 권한이 있는 NVMe/TCP용 기본 PVC를 보여 줍니다 protection-gold.

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

POD 매니페스트 샘플

이 예는 PVC를 포드에 부착하기 위한 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
```

Trident에서 볼륨을 프로비저닝하는 방법을 제어하는 데 사용되는 및 매개 변수와 스토리지 클래스가 상호 작용하는 방법에 대한 자세한 PersistentVolumeClaim 내용은 ["Kubernetes 및 Trident 오브젝트"](#)참조하십시오.

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 볼륨을 확장합니다

CSI 프로비저닝을 사용하여 iSCSI PV(Persistent Volume)를 확장할 수 있습니다.



iSCSI 볼륨 확장은 ONTAP-SAN, ONTAP-SAN-이코노미, Solidfire-SAN 드라이버로 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass**를 구성합니다

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

```
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` 매개변수를 포함하도록 편집합니다.

2단계: 생성한 **StorageClass**를 사용하여 **PVC**를 생성합니다

PVC 정의를 편집하고 `spec.resources.requests.storage` 원래 크기보다 커야 하는 새로 원하는 크기를 반영합니다.

```
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 만들고 이 영구 볼륨 클레임(PVC)과 연결합니다.

```

kubect1 get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO           ontap-san    8s

kubect1 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: PVC를 부착하는 POD를 정의합니다

크기를 조정할 수 있도록 PV를 포드에 연결합니다. iSCSI PV의 크기를 조정할 때 두 가지 시나리오가 있습니다.

- PV가 Pod에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 하면 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 POD에 바인딩되면 Trident가 디바이스를 다시 검사해 파일 시스템의 크기를 조정합니다. 그런 다음 확장 작업이 성공적으로 완료된 후 Kubernetes에서 PVC 크기를 업데이트합니다.

이 예제에서는 'AN-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: PV를 확장합니다

1Gi 에서 2Gi 로 생성된 PV 의 크기를 조정하려면 PVC 정의를 편집하여 'pec.resources.requests.storage'를 2Gi 로 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 확인

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

```

kubect1 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
kubect1 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 볼륨을 확장합니다

CSI Provisioner를 사용하여 FC Persistent Volume(PV)을 확장할 수 있습니다.



FC 볼륨 확장은 `ontap-san` 드라이버에서 지원되며 Kubernetes 1.16 이상이 필요합니다.

1단계: 볼륨 확장을 지원하도록 **StorageClass**를 구성합니다

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

```
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 매개변수를 포함하도록 편집합니다.

2단계: 생성한 **StorageClass**를 사용하여 **PVC**를 생성합니다

PVC 정의를 편집하고 를 업데이트합니다 spec.resources.requests.storage 원래 크기보다 커야 하는 새로 원하는 크기를 반영합니다.

```
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 만들고 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 부착하는 **POD**를 정의합니다

크기를 조정할 수 있도록 PV를 포드에 연결합니다. FC PV 크기를 조정할 때는 두 가지 시나리오가 있습니다.

- PV가 Pod에 연결된 경우 Trident는 스토리지 백엔드에서 볼륨을 확장하고 디바이스를 다시 스캔하고 파일 시스템의 크기를 조정합니다.
- 연결되지 않은 PV의 크기를 조정하려고 하면 Trident는 스토리지 백엔드에서 볼륨을 확장합니다. PVC가 POD에 바인딩되면 Trident가 디바이스를 다시 검사해 파일 시스템의 크기를 조정합니다. 그런 다음 확장 작업이 성공적으로 완료된 후 Kubernetes에서 PVC 크기를 업데이트합니다.

이 예제에서는 'AN-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: PV를 확장합니다

1Gi 에서 2Gi 로 생성된 PV 의 크기를 조정하려면 PVC 정의를 편집하여 'pec.resources.requests.storage'를 2Gi 로 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 확인

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

```

kubect1 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
kubect1 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 볼륨을 확장합니다

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

1단계: 볼륨 확장을 지원하도록 **StorageClass**를 구성합니다

NFS PV의 크기를 조정하려면 먼저 관리자가 "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 스토리지 클래스를 이미 생성한 경우 'kubect edit storageclass'를 사용하여 기존 스토리지 클래스를 편집하여 볼륨을 확장할 수 있습니다.

2단계: 생성한 **StorageClass**를 사용하여 **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**를 확장합니다

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: 확장을 확인합니다

PVC, PV 및 Trident 볼륨의 크기를 확인하여 크기가 올바르게 조정되었는지 확인할 수 있습니다.

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb      Bound       pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO                 ontapnas          4m44s

kubect1 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 26.02에서 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.

- san
- nas
- san-nvme
- google-cloud-netapp-volumes

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
 - 인터넷 소형 컴퓨터 시스템 인터페이스
 - Fibre Channel 프로토콜
 - 패브릭을 통한 Non-Volatile Memory Express

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
 - 인터넷 소형 컴퓨터 시스템 인터페이스
 - Fibre Channel 프로토콜
 - 패브릭을 통한 Non-Volatile Memory Express

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.

필드에 입력합니다	설명	형식	필수 요소입니다	예	기본값
이름	고유 정책 식별자	문자열	예	production-db-policy	None
usedThreshhold	확장을 트리거하는 용량 비율	백분율 문자열	예	"80%"	None

필드에 입력합니다	설명	형식	필수 요소입니다	예	기본값
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.
실패했습니다	유효성 검사 오류가 감지되었습니다.	사양을 검토하고 수정하십시오.

상태	설명	조치 필요
삭제 중	삭제가 진행 중입니다.	완료될 때까지 기다립니다.

정책을 **StorageClass**에 연결합니다

`trident.netapp.io/autogrowPolicy` 어노테이션을 사용하여 **StorageClass**에 자동 확장 정책을 연결할 수 있습니다. 해당 **StorageClass**에서 프로비저닝된 모든 볼륨은 이 정책을 상속받습니다.



StorageClass는 다음을 가져야 합니다 `allowVolumeExpansion: true`.

단계

1. Autogrow Policy 어노테이션을 사용하여 **StorageClass**를 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. **StorageClass**를 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
```

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

Trident로 볼륨을 가져올 수 있는 대상:

- 응용 프로그램을 Containerize 하고 기존 데이터 집합을 다시 사용합니다
- 수명이 짧은 애플리케이션에 사용할 데이터 세트의 클론을 사용합니다
- 오류가 발생한 Kubernetes 클러스터를 재구성합니다
- 재해 복구 중에 애플리케이션 데이터 마이그레이션

고려 사항

볼륨을 가져오기 전에 다음 고려 사항을 검토하십시오.

- Trident는 RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 대상 볼륨입니다. 볼륨을 Trident로 가져오기 전에 미리 관계를 해제해야 합니다.
- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 활성 볼륨을 가져오려면 볼륨을 클론한 다음 가져오기를 수행합니다.



Kubernetes가 이전 연결을 인식하지 못하고 활성 볼륨을 POD에 쉽게 연결할 수 있기 때문에 블록 볼륨에서 특히 중요합니다. 이로 인해 데이터가 손상될 수 있습니다.

- PVC에 지정해야 하지만 StorageClass Trident는 가져오는 동안 이 매개변수를 사용하지 않습니다. 스토리지 클래스는 볼륨 생성 중에 스토리지 특성에 따라 사용 가능한 풀에서 선택하는 데 사용됩니다. 볼륨이 이미 있으므로 가져오는 동안 풀을 선택할 필요가 없습니다. 따라서 볼륨이 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드 또는 풀에 있더라도 가져오기에 실패합니다.
- 기존 체적 크기는 PVC에서 결정되고 설정됩니다. 스토리지 드라이버에서 볼륨을 가져온 후 PV는 PVC에 대한 ClaimRef를 사용하여 생성됩니다.
 - 처음에 부가세 반환 청구액 정책이 로 설정되어 있습니다 retain PV에서 Kubernetes에서 PVC 및 PV를 성공적으로 바인딩하면 스토리지 클래스의 부가세 반환 청구액 정책에 맞게 부가세 반환 청구액 정책이 업데이트됩니다.
 - 스토리지 클래스의 부가세 반환 청구액 정책이 인 경우 delete, PV 삭제 시 저장 볼륨이 삭제된다.
- 기본적으로 Trident PVC를 관리하고 백엔드에서 FlexVol volume 과 LUN의 이름을 변경합니다. 당신은 통과 할 수 있습니다 --no-manage 관리되지 않는 볼륨을 가져오기 위한 플래그 및 --no-rename 볼륨 이름을 유지하기 위한 플래그입니다.
 - --no-manage* - 사용하는 경우 --no-manage 플래그가 지정되면 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.
 - --no-rename* - 사용하는 경우 --no-rename 플래그를 사용하면 Trident 볼륨을 가져오는 동안 기존 볼륨 이름을 유지하고 볼륨의 수명 주기를 관리합니다. 이 옵션은 다음에 대해서만 지원됩니다. ontap-nas , ontap-san (ASA r2 시스템 포함) 및 ontap-san-economy 운전자.



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

- PVC 및 PV에 주석이 추가되어 용적을 가져온 후 PVC와 PV가 관리되었는지 여부를 나타내는 두 가지 목적으로 사용됩니다. 이 주석은 수정하거나 제거할 수 없습니다.

볼륨을 가져옵니다

``tridentctl import``를 사용하거나 Trident 가져오기 주석이 포함된 PVC를 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)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.

예

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하십시오.

ONTAP NAS 및 ONTAP NAS FlexGroup를 지원합니다

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



- Trident 다음을 사용하여 볼륨 가져오기를 지원하지 않습니다. `ontap-nas-economy` 운전사.
- 를 클릭합니다 `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버는 중복 볼륨 이름을 허용하지 않습니다.

드라이버로 생성된 각 볼륨은 `ontap-nas` ONTAP 클러스터의 FlexVol volume입니다. 드라이버로 FlexVol 볼륨을 가져오는 `ontap-nas` 작업은 동일합니다. ONTAP 클러스터에 이미 있는 FlexVol 볼륨을 PVC로 가져올 수 `ontap-nas` 있습니다. 마찬가지로 FlexGroup 볼륨을 PVC로 가져올 수 `ontap-nas-flexgroup` 있습니다.

tridentctl을 사용한 ONTAP NAS 예제

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

관리 볼륨

다음 예에서는 라는 볼륨을 가져옵니다 managed_volume 백엔드에서 을(를) 선택합니다 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

인수를 사용할 때 --no-manage Trident는 볼륨의 이름을 바꾸지 않습니다.

다음 예에서는 를 가져옵니다 unmanaged_volume 를 누릅니다 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.

관리 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 1GiB `ontap-nas` 볼륨을 가져옵니다.

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

ONTAP SAN

Trident 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.

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리 볼륨

관리되는 볼륨의 경우 Trident의 이름은 FlexVol volume의 이름을 형식으로, FlexVol volume 내의 LUN의 lun0 이름은 pvc-<uuid> 으로 바꿉니다.

다음 예에서는 ontap-san-managed 백엔드에 있는 FlexVol volume를 ontap_san_default 가져옵니다.

```
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 예에서는 를 가져옵니다 unmanaged_example_volume 를 누릅니다 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 예에 표시된 것처럼 IQN을 Kubernetes 노드 IQN과 공유하는 igroup에 LUN이 매핑되어 있는 경우 오류가 발생합니다. LUN already mapped to initiator(s) in this group. 볼륨을 가져오려면 이니시에이터를 제거하거나 LUN 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

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



Element 드라이버는 중복 볼륨 이름을 지원합니다. 그러나 볼륨 이름이 중복되면 Trident에서 오류를 반환합니다. 이 문제를 해결하려면 볼륨을 클론하고 고유한 볼륨 이름을 제공한 다음 복제된 볼륨을 가져옵니다.

다음 예제에서는 을 가져옵니다 element-managed 백엔드의 볼륨 element_default.

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

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
block	pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe	d3ba047a-ea0b-43f9-9c42-e38e58301c49	10 GiB	online	basic-element	true

Azure NetApp Files

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



Azure NetApp Files 볼륨을 가져오려면 해당 볼륨 경로를 기준으로 볼륨을 식별합니다. 볼륨 경로는 이후 볼륨 내보내기 경로의 일부입니다 :/. 예를 들어, 마운트 경로가 인 경우 10.0.0.2:/importvol1, 볼륨 경로는 입니다 importvol1.

다음 예제에서는 을 가져옵니다 azure-netapp-files 백엔드의 볼륨 azurenetappfiles_40517 볼륨 경로 포함 importvol1.

```
tridentctl import volume azurenetappfiles_40517 importvoll1 -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 볼륨

Trident는 드라이버를 사용하여 볼륨 가져오기를 `google-cloud-netapp-volumes` 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 예에서는 동일한 영역에 두 개의 볼륨이 있을 때 볼륨을 가져옵니다 `google-cloud-netapp-volumes`.

```
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 레이블을 사용자 지정합니다

Trident를 사용하면 생성한 볼륨에 의미 있는 이름과 레이블을 할당할 수 있습니다. 이를 통해 볼륨을 해당 Kubernetes 리소스(PVC)에 식별하고 쉽게 매핑할 수 있습니다. 백엔드 레벨에서 사용자 지정 볼륨 이름 및 사용자 지정 레이블을 생성하기 위한 템플릿을 정의할 수도 있습니다. 생성, 가져오기 또는 복제하는 모든 볼륨은 템플릿에 적용됩니다.

시작하기 전에

사용자 지정 가능한 볼륨 이름 및 레이블 지원:

- 볼륨 생성, 가져오기 및 클론 복제 작업입니다.
- 의 경우에는 `ontap-nas-economy` 드라이버의 경우 Qtree 볼륨의 이름만 이름 템플릿을 따릅니다.
- 의 경우에는 `ontap-san-economy` 드라이버의 경우 LUN 이름만 이름 템플릿을 준수합니다.

제한 사항

- 사용자 정의 볼륨 이름은 ONTAP 온프레미스 드라이버와만 호환됩니다.
- 사용자 정의 레이블은 다음에 대해서만 지원됩니다. `ontap-san`, `ontap-nas`, 그리고 `ontap-nas-flexgroup` 운전자.
- 사용자 정의 볼륨 이름은 기존 볼륨에는 적용되지 않습니다.

사용자 지정 가능한 볼륨 이름의 주요 동작

- 이름 템플릿의 잘못된 구문으로 인해 오류가 발생하면 백엔드 생성이 실패합니다. 그러나 템플릿 응용 프로그램이 실패하면 볼륨의 이름은 기존 명명 규칙에 따라 지정됩니다.
- 백엔드 구성에서 이름 템플릿을 사용하여 볼륨 이름을 지정한 경우에는 스토리지 접두사를 사용할 수 없습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블이 있는 백엔드 구성 예

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

루트 수준의 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.RequestName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

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

템플릿 예제를 명명합니다

- 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. 볼륨을 가져오는 경우 기존 볼륨에 특정 형식의 레이블이 있는 경우에만 레이블이 업데이트됩니다. 예를 들면 다음과 같이 `{"provisioning":{"Cluster":"ClusterA"}, "PVC": "pvname"}` 같습니다.
2. 관리되는 볼륨을 가져오는 경우 볼륨 이름은 백엔드 정의의 루트 레벨에 정의된 이름 템플릿을 따릅니다.
3. Trident는 스토리지 접두사가 있는 슬라이스 연산자 사용을 지원하지 않습니다.
4. 템플릿으로 인해 고유한 볼륨 이름이 생성되지 않을 경우 Trident는 몇 개의 임의 문자를 추가하여 고유한 볼륨 이름을 생성합니다.
5. NAS 경제 볼륨의 사용자 지정 이름이 64자를 초과하는 경우 Trident는 기존 명명 규칙에 따라 볼륨의 이름을 지정합니다. 다른 모든 ONTAP 드라이버의 경우 볼륨 이름이 이름 제한을 초과하면 볼륨 생성 프로세스가 실패합니다.

네임스페이스 전체에서 NFS 볼륨을 공유합니다

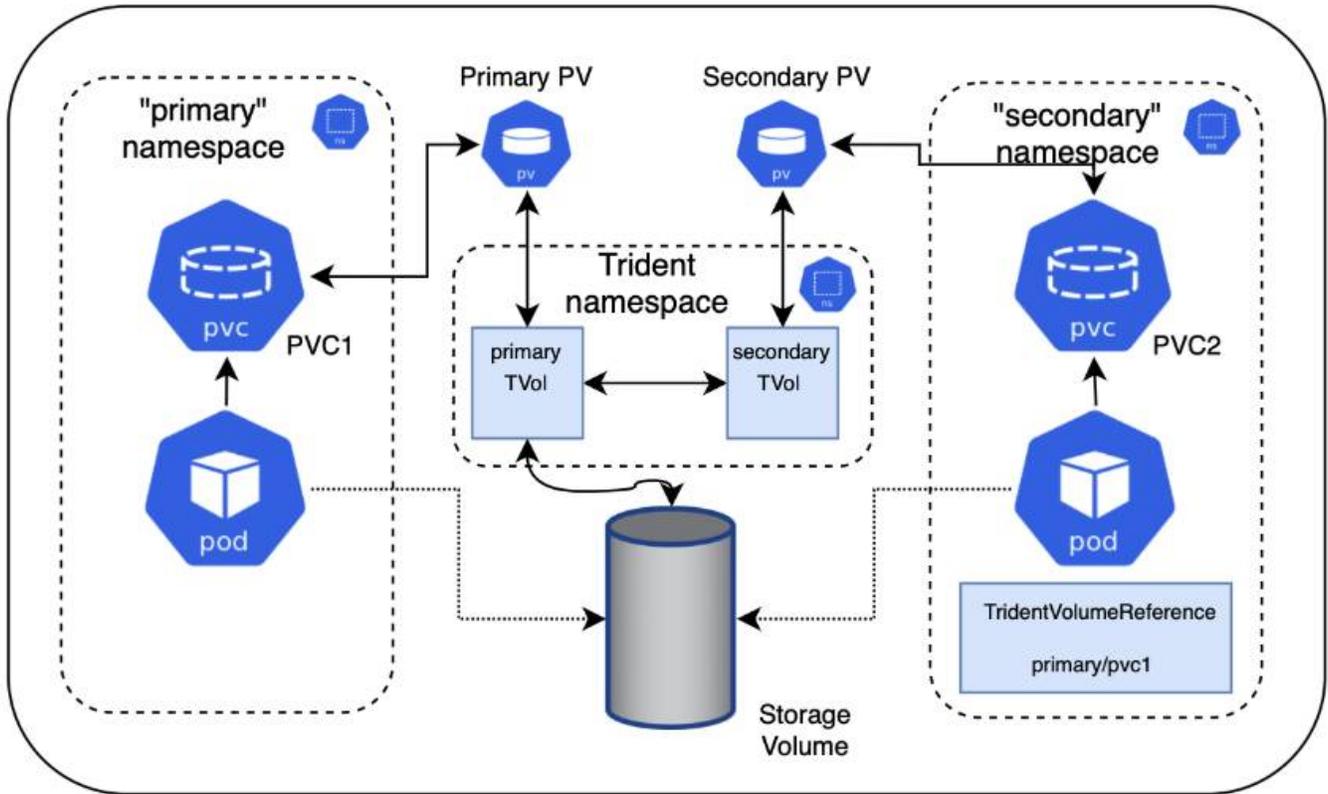
Trident를 사용하면 기본 네임스페이스에 볼륨을 생성한 후 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

피처

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 다양한 수준의 액세스 제어
- 모든 Trident NFS 볼륨 드라이버와 호환됩니다
- `tridentctl` 또는 기타 기본 Kubernetes 기능이 아닌 기능에 의존하지 않습니다

이 다이어그램은 2개의 Kubernetes 네임스페이스에서 NFS 볼륨 공유를 보여 줍니다.



빠른 시작

몇 단계만으로 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 네임스페이스에서 **CR**을 만들 수 있는 권한을 부여합니다

클러스터 관리자는 대상 네임스페이스의 소유자에게 트리엔VolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3

대상 네임스페이스에서 트리엔**VolumeReference** 를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 트리엔VolumeReference CR을 생성합니다.

4

대상 네임스페이스에서 하위 **PVC**를 만듭니다

대상 네임스페이스의 소유자는 원본 PVC의 데이터 소스를 사용하기 위해 하위 PVC를 만듭니다.

소스 및 대상 네임스페이스를 구성합니다

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

단계

1. * 원본 네임스페이스 소유자: * PVC를 만듭니다 (pvc1)를 대상 네임스페이스와 공유할 수 있는 권한을 부여하는 소스 네임스페이스의 경우 (namespace2)를 사용합니다 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
```

Trident은 PV와 백엔드 NFS 스토리지 볼륨을 생성합니다.



- 쉽표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. 예를 들면, 다음과 같습니다. trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4.
- 를 사용하여 모든 네임스페이스에 공유할 수 있습니다 *. 예를 들면, 다음과 같습니다. trident.netapp.io/shareToNamespace: *
- PVC를 업데이트하여 를 포함할 수 있습니다 shareToNamespace 언제든지 주석을 추가할 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 네임스페이스에서 트리젠VolumeReference CR을 만듭니다 pvc1.

```

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

```

4. * 대상 네임스페이스 소유자: * PVC를 만듭니다 (pvc2)를 대상 네임스페이스에서 사용합니다 (namespace2)를 사용합니다 shareFromPVC 원본 PVC를 지정하는 주식.

```

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 주석을 읽고 shareFromPVC 대상 PV를 소스 PV를 가리키고 소스 PV 스토리지 리소스를 공유하는 자체 스토리지 리소스가 없는 하위 볼륨으로 대상 PV를 생성합니다. 대상 PVC와 PV가 정상으로 표시됩니다.

공유 볼륨을 삭제합니다

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

사용 tridentctl get 하위 볼륨을 쿼리합니다

를 사용합니다[tridentctl 유틸리티, 를 실행할 수 있습니다 get 하위 볼륨을 가져오는 명령입니다. 자세한 내용은 다음 링크를 참조하십시오.../trident-reference/tridentctl.html[tridentctl 명령 및 옵션].

Usage:

```
tridentctl get [option]
```

플래그:

- `-h, --help`: 볼륨에 대한 도움말입니다.
- `--parentOfSubordinate string`: 하위 원본 볼륨으로 쿼리를 제한합니다.
- `--subordinateOf string`: 볼륨 부하로 쿼리 제한.

제한 사항

- Trident는 대상 네임스페이스가 공유 볼륨에 쓰는 것을 방지할 수 없습니다. 파일 잠금 또는 기타 프로세스를 사용하여 공유 볼륨 데이터를 덮어쓰지 않도록 해야 합니다.
- 를 제거하여 원본 PVC에 대한 액세스를 취소할 수 없습니다 `shareToNamespace` 또는 `shareFromNamespace` 주식 또는 삭제 `TridentVolumeReference` 있습니다. 액세스 권한을 취소하려면 하위 PVC를 삭제해야 합니다.
- 하위 볼륨에서는 스냅샷, 클론 및 미러링을 사용할 수 없습니다.

를 참조하십시오

네임스페이스 간 볼륨 액세스에 대한 자세한 내용은 다음을 참조하십시오.

- 를 방문하십시오 ["네임스페이스 간 볼륨 공유: 네임스페이스 간 볼륨 액세스를 위해 hello를 사용합니다"](#).
- 데모를 시청해보시기 ["NetAppTV를 참조하십시오"](#)바랍니다.

네임스페이스 전체에 걸친 클론 볼륨

Trident를 사용하면 동일한 Kubernetes 클러스터 안에 있는 다른 네임스페이스의 기존 볼륨 또는 볼륨스냅샷을 사용하여 새 볼륨을 생성할 수 있습니다.

필수 구성 요소

볼륨을 클론 복제하기 전에 소스 및 대상 백엔드의 유형이 동일하고 스토리지 클래스가 동일인지 확인합니다.



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.



볼륨을 복제하도록 소스 **PVC**를 구성합니다

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

2

대상 네임스페이스에서 **CR**을 만들 수 있는 권한을 부여합니다

클러스터 관리자는 대상 네임스페이스의 소유자에게 트리엔VolumeReference CR을 생성할 수 있는 권한을 부여합니다.

3

대상 네임스페이스에서 트리엔**VolumeReference** 를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 트리엔VolumeReference CR을 생성합니다.

4

대상 네임스페이스에 클론 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 PVC를 생성하여 소스 네임스페이스에서 PVC를 복제합니다.

소스 및 대상 네임스페이스를 구성합니다

보안을 보장하기 위해 네임스페이스 전체에서 볼륨을 클론 복제하려면 소스 네임스페이스 소유자, 클러스터 관리자 및 대상 네임스페이스 소유자의 협업과 작업이 필요합니다. 사용자 역할은 각 단계에서 지정됩니다.

단계

1. * 소스 네임스페이스 소유자: * (pvc1`소스 네임스페이스에 PVC 생성 (`namespace1)(namespace2. 대상 네임스페이스와 공유할 수 있는 권한을 부여함) 주석을 사용합니다. cloneToNamespace

```

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

```

Trident은 PV와 백엔드 스토리지 볼륨을 생성합니다.



- 심표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` 예를 들어,
- 을 사용하여 모든 네임스페이스에 공유할 수 * 있습니다. 예를 들면, 다음과 같습니다.
trident.netapp.io/cloneToNamespace: *
- 주석을 포함하도록 PVC를 업데이트할 수 cloneToNamespace 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 네임스페이스에서 트리젠VolumeReference CR을 만듭니다 pvc1.

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

4. * 대상 네임스페이스 소유자: * (pvc2`대상 네임스페이스에서 PVC 생성 (`namespace2) 또는 cloneFromSnapshot, cloneFromNamespace 주석을 사용하여 cloneFromPVC 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-이코노미 드라이버를 사용하여 프로비저닝된 PVC의 경우 읽기 전용 클론은 지원되지 않습니다.

SnapMirror를 사용하여 볼륨을 복제합니다

Trident은 재해 복구에 대비해 데이터를 복제하기 위해 한 클러스터의 소스 볼륨과 피어링된 클러스터의 타겟 볼륨 간의 미러링 관계를 지원합니다. Trident Mirror Relationship(TMR)이라고 하는 네임스페이스가 지정된 사용자 지정 리소스 정의(CRD)를 사용하여 다음 작업을 수행할 수 있습니다.

- 볼륨 간 미러 관계 생성(PVC)
- 볼륨 간 미러 관계를 제거합니다
- 미러 관계를 해제합니다
- 재해 상태(페일오버) 중에 2차 볼륨 승격
- 계획된 페일오버 또는 마이그레이션 중에 클러스터에서 클러스터로 애플리케이션의 무손실 전환 수행

복제 사전 요구 사항

시작하기 전에 다음과 같은 사전 요구 사항이 충족되는지 확인하십시오.

ONTAP 클러스터

- * Trident *: Trident 버전 22.10 이상이 ONTAP를 백엔드로 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 있어야 합니다.
- * 라이선스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스를 활성화해야 합니다. 자세한 내용은 을 ["ONTAP의 SnapMirror 라이선스 개요"](#) 참조하십시오.

ONTAP 9.10.1부터 모든 라이선스는 여러 기능을 사용할 수 있는 단일 파일인 NetApp 라이선스 파일(NLF)로 제공됩니다. 자세한 내용은 을 ["ONTAP One에 포함된 라이선스"](#) 참조하십시오.



SnapMirror 비동기 보호만 지원됩니다.

피어링

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 자세한 내용은 을 ["클러스터 및 SVM 피어링 개요"](#) 참조하십시오.



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

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

지원되는 드라이버

NetApp Trident는 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)



SnapMirror를 사용한 볼륨 복제는 ASA r2 시스템에서는 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 다음을 참조하세요. ["ASA R2 스토리지 시스템에 대해 알아보십시오"](#) .

대칭 복사된 PVC를 작성합니다

다음 단계를 수행하고 CRD 예제를 사용하여 운영 볼륨과 보조 볼륨 간의 미러 관계를 생성합니다.

단계

1. 운영 Kubernetes 클러스터에서 다음 단계를 수행합니다.

a. 매개 변수를 사용하여 StorageClass 개체를 `trident.netapp.io/replication: true` 만듭니다.

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 생성된 StorageClass를 사용하여 PVC를 생성합니다.

예

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

c. 로컬 정보로 MirrorRelationship CR을 생성합니다.

예

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

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

- d. TridentMirrorRelationship CR을 가져와 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 매개 변수를 사용하여 StorageClass 를 만듭니다.

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"
```

Trident에서 구성된 관계 정책 이름(또는 ONTAP의 경우 기본값)을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

- c. 이전에 생성한 StorageClass를 사용하여 PVC를 생성하여 보조(SnapMirror 대상) 역할을 합니다.

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 없는 경우 볼륨을 생성하지 못합니다. 이 관계가 있으면 Trident은 새로운 FlexVol volume을 MirrorRelationship에 정의된 원격 SVM과 피어링된 SVM에 배치하도록 보장합니다.

볼륨 복제 상태입니다

Trident Mirror Relationship(TMR)은 PVC 간 복제 관계의 한쪽 끝을 나타내는 CRD입니다. 대상 TMR에는 원하는 상태를 Trident에 알려주는 상태가 있습니다. 대상 TMR의 상태는 다음과 같습니다.

- * 설립 * : 로컬 PVC는 미리 관계의 대상 볼륨이며, 이것은 새로운 관계입니다.
- * 승진된 * : 로컬 PVC는 현재 유효한 미리 관계가 없는 ReadWrite 및 마운트 가능합니다.
- * 재설립 * : 로컬 PVC는 미리 관계의 대상 볼륨이며 이전에 해당 미리 관계에 있었습니다.
 - 대상 볼륨이 대상 볼륨 내용을 덮어쓰므로 대상 볼륨이 소스 볼륨과 관계가 있는 경우 다시 설정된 상태를 사용해야 합니다.
 - 볼륨이 소스와 이전에 관계가 없는 경우 재설정된 상태가 실패합니다.

비계획 파일오버 중에 보조 PVC를 승격합니다

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 `_spec.state_field`를 로 `'promoted'` 업데이트합니다.

계획된 파일오버 중에 보조 PVC를 승격합니다

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

단계

1. 운영 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 운영 Kubernetes 클러스터에서 SnapshotInfo CR을 생성하여 내부 세부 정보를 가져옵니다.

예

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

3. 보조 Kubernetes 클러스터에서 `TridentMirrorRelationship_CR`의 `_spec.state_field`를 `_promitted` 및 `spec.promotedSnapshotHandle` 으로 업데이트하여 스냅샷의 내부 이름으로 업데이트합니다.
4. 보조 Kubernetes 클러스터에서 승격될 TridentMirrorRelationship의 상태(status.state 필드)를 확인합니다.

파일오버 후 미러 관계를 복구합니다

미러 관계를 복구하기 전에 새 1차 사이트로 만들 측면을 선택합니다.

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 `_spec.remoteVolumeHandle_field` 값이 업데이트되었는지 확인합니다.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 `_spec.mirror_field`를 로 ``reestablished`` 업데이트합니다.

추가 작업

Trident는 1차 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

1차 PVC를 새로운 2차 PVC로 복제합니다

이미 1차 PVC와 2차 PVC가 있는지 확인하십시오.

단계

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

대칭 복사, 1차 또는 2차 PVC의 크기를 조정합니다

PVC는 평소대로 크기를 조정할 수 있으며, 데이터 양이 현재 크기를 초과할 경우 ONTAP는 자동으로 대상 flexvols를 확장합니다.

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에 대해 다음 작업 중 하나를 수행합니다.

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

PVC 삭제(이전에 미러링됨)

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

TMR을 삭제합니다

미러링된 관계의 한 쪽에서 TMR을 삭제하면 Trident에서 삭제를 완료하기 전에 나머지 TMR이 `_PROJED_STATE`로 전환됩니다. 삭제하도록 선택한 TMR이 이미 `_PROJED_STATE`에 있는 경우 기존 미러 관계가 없으며 TMR이 제거되고 Trident가 로컬 PVC를 `_ReadWrite_`로 승격합니다. 이렇게 삭제하면 ONTAP의 로컬 볼륨에 대한 SnapMirror 메타데이터가 해제됩니다. 이 볼륨이 향후 미러 관계에 사용될 경우 새 미러 관계를 생성할 때 `_established_volume` 복제 상태의 새 TMR을 사용해야 합니다.

ONTAP가 온라인 상태일 때 미리 관계를 업데이트합니다

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

ONTAP이 오프라인일 때 미리 관계를 업데이트합니다

Trident이 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 반영합니다. 이 값은 *SUCCEEDED*, *In Progress* 또는 *_Failed_*에서 가져올 수 있습니다.

CSI 토폴로지를 사용합니다

Trident은 를 사용하여 Kubernetes 클러스터에 있는 노드에 볼륨을 선택적으로 생성하고 연결할 수 있습니다 "[CSI 토폴로지 기능](#)".

개요

CSI 토폴로지 기능을 사용하면 지역 및 가용성 영역에 따라 볼륨에 대한 액세스가 노드의 하위 집합으로 제한될 수 있습니다. 오늘날의 클라우드 공급자는 Kubernetes 관리자가 영역 기반의 노드를 생성할 수 있습니다. 노드는 지역 내 또는 여러 지역의 여러 가용성 영역에 위치할 수 있습니다. Trident는 다중 영역 아키텍처에서 워크로드용 볼륨을 간편하게 프로비저닝할 수 있도록 CSI 토폴로지를 사용합니다.



CSI 토폴로지 기능에 대해 자세히 알아보십시오 "[여기](#)".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- `VolumeBindingMode`로 `Immediate` 설정하면 Trident에서 토폴로지를 인식하지 않고 볼륨을 생성합니다. 볼륨 바인딩 및 동적 프로비저닝은 PVC가 생성될 때 처리됩니다. 이 옵션은 기본값이며 `VolumeBindingMode` 토폴로지 제약 조건을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청 Pod의 예약 요구사항에 종속되지 않고 생성됩니다.
- `VolumeBindingMode`를 `WaitForFirstConsumer`로 설정하면 PVC를 사용하는 POD가 예약 및 생성될 때까지 PVC에 대한 영구 볼륨의 생성 및 바인딩이 지연됩니다. 이렇게 하면 토폴로지 요구 사항에 따라 적용되는 일정 제한을 충족하기 위해 볼륨이 생성됩니다.



WaitForFirstConsumer의 바인딩 모드에는 토폴로지 레이블이 필요하지 않습니다. 이 기능은 CSI 토폴로지 기능과 독립적으로 사용할 수 있습니다.

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- 를 실행하는 Kubernetes 클러스터 ["지원되는 Kubernetes 버전"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
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:"1e11e4a2108024935ecfcb2912226cedeadfd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 클러스터의 노드에는 토폴로지 인식 및 `topology.kubernetes.io/zone` 을 설명하는 레이블이 있어야 (`topology.kubernetes.io/region``합니다. 토폴로지를 인식하려면 Trident를 설치하기 전에 클러스터의 노드에 이러한 레이블 * 이 있어야 Trident 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","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

1단계: 토폴로지 인식 백엔드 생성

Trident 스토리지 백엔드는 가용 영역을 기준으로 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드에는 지원되는 영역 및 영역 목록을 나타내는 선택적 블록이 포함될 수 `supportedTopologies` 있습니다. 이러한 백엔드를 사용하는 `StorageClasses`의 경우 지원되는 영역/영역에서 예약된 애플리케이션에서 요청하는 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

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
```

JSON을 참조하십시오

```
{
  "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 백엔드당 지역 및 영역 목록을 제공하는 데 사용됩니다. 이러한 영역 및 영역은 StorageClass 에서 제공할 수 있는 허용 가능한 값의 목록을 나타냅니다. 백엔드에서 제공되는 영역 및 영역의 하위 집합이 포함된 StorageClasses의 경우 Trident는 백엔드에 볼륨을 생성합니다.

스토리지 풀별로 'SupportedTopologies'를 정의할 수도 있습니다. 다음 예를 참조하십시오.

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-a
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall
        topology.kubernetes.io/zone: us-centrall-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall
        topology.kubernetes.io/zone: us-centrall-b

```

이 예에서는 "reGion" 및 "zone" 레이블이 스토리지 풀의 위치를 나타냅니다. "topology.Kubernetes.io/region" 및 "topology.Kubernetes.io/zone"은 스토리지 풀을 사용할 수 있는 위치를 지정합니다.

2단계: 토폴로지를 인식하는 **StorageClasses**를 정의합니다

클러스터의 노드에 제공되는 토폴로지 레이블을 기반으로 StorageClasses를 정의하여 토폴로지 정보를 포함할 수 있습니다. 이렇게 하면 PVC 요청에 대한 후보 역할을 하는 스토리지 풀과 Trident에서 제공하는 볼륨을 사용할 수 있는 노드의 하위 세트가 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 로 WaitForFirstConsumer` 설정됩니다. 이 StorageClass에 요청된 PVC는 POD에서 참조될 때까지 작동하지 않습니다. 및 는 `allowedTopologies 사용할 영역 및 영역을 제공합니다. netapp-san-us-east1`StorageClass는 위에 정의된 백엔드에 PVC를 `san-backend-us-east1 생성합니다.

3단계: PVC 생성 및 사용

StorageClass가 생성되어 백엔드에 매핑되면 PVC를 생성할 수 있습니다.

아래의 '샘플'을 참조하십시오.

```

---
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 만들면 다음과 같은 결과가 발생합니다.

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY      ACCESS MODES      STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age   From
  ----      -
  Normal    WaitForFirstConsumer 6s    persistentvolume-controller
waiting
for first consumer to be created before binding

```

Trident에서 볼륨을 생성하여 PVC에 바인딩하려면 POD에서 PVC를 사용합니다. 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: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

이 podSpec은 us-east1 지역에 존재하는 노드에서 pPod를 예약하고 us-east1-a 또는 us-east1-b 영역에 있는 노드 중에서 선택하도록 지시합니다.

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
```

포함할 백엔드를 업데이트합니다 supportedTopologies

기존 백엔드는 'tridentctl backend update'를 사용하여 'supportedTopologies' 목록을 포함하도록 업데이트할 수 있습니다. 이는 이미 프로비저닝된 체적에 영향을 주지 않으며 후속 PVC에만 사용됩니다.

자세한 내용을 확인하십시오

- ["컨테이너에 대한 리소스를 관리합니다"](#)
- ["노드 선택기"](#)
- ["친화성 및 반친화성"](#)
- ["오염과 내약입니다"](#)

스냅샷 작업

영구 볼륨(PVS)의 Kubernetes 볼륨 스냅샷은 볼륨의 시점 복사본을 지원합니다. 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 다음에서 지원됩니다. `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files`, 그리고 `google-cloud-netapp-volumes` 운전자.

시작하기 전에

스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. Kubernetes Orchestrator의 책임입니다(예: Kubeadm, GKE, OpenShift).

Kubernetes 배포 시 스냅샷 컨트롤러 및 CRD가 포함되지 않은 경우 를 참조하십시오 [볼륨 스냅샷 컨트롤러를 배포합니다](#).



GKE 환경에서 필요 시 볼륨 스냅샷을 생성할 경우 스냅샷 컨트롤러를 생성하지 마십시오. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

볼륨 스냅샷을 생성합니다

단계

1. 을 생성합니다 VolumeSnapshotClass. 자세한 내용은 을 참조하십시오 ["VolumeSnapshotClass"](#).
 - 가 driver Trident CSI 드라이버를 가리킵니다.
 - deletionPolicy 있을 수 있습니다 Delete 또는 Retain. 를 로 설정한 경우 Retain, 스토리지 클러스터의 기본 물리적 스냅샷은 가 있는 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 생성합니다.

예

- 이 예에서는 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 예에서는 라는 PVC에 대한 볼륨 스냅샷 객체를 생성합니다 pvc1 스냅샷 이름이 로 설정되어 있습니다 pvc1-snap. VolumeSnapshot은 PVC와 유사하며 와 관련이 있습니다 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
```

- 를 식별할 수 있습니다 VolumeSnapshotContent 의 개체 pvc1-snap VolumeSnapshot을 설명합니다. 를 클릭합니다 Snapshot Content Name 이 스냅샷을 제공하는 VolumeSnapshotContent 객체를 식별합니다. 를 클릭합니다 Ready To Use 매개 변수는 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 PVC를 생성합니다

을 사용할 수 있습니다 dataSource 이름이 인 VolumeSnapshot을 사용하여 PVC를 생성합니다 <pvc-name> 데이터 소스로 사용됩니다. PVC가 생성된 후 POD에 부착하여 다른 PVC와 마찬가지로 사용할 수 있습니다.



PVC는 소스 볼륨과 동일한 백엔드에서 생성됩니다. 을 참조하십시오 ["KB: Trident PVC 스냅샷에서 PVC를 생성하는 것은 대체 백엔드에서 생성할 수 없습니다"](#).

다음 예에서는 를 사용하여 PVC를 작성합니다 pvc1-snap 를 데이터 소스로 사용합니다.

```
cat pvc-from-snap.yaml
```

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

볼륨 스냅샷을 가져옵니다

Trident는 클러스터 관리자가 객체를 생성하고 Trident 외부에서 생성된 스냅샷을 가져올 수 있도록 하기 위해 VolumeSnapshotContent 를 ["Kubernetes 사전 프로비저닝된 스냅샷 프로세스"](#) 지원합니다.

시작하기 전에

Trident에서 스냅샷의 상위 볼륨을 생성하거나 가져와야 합니다.

단계

1. * 클러스터 관리자: * VolumeSnapshotContent 백엔드 스냅샷을 참조하는 객체를 생성합니다. 그러면 Trident에서 스냅샷 워크플로우가 시작됩니다.

- 에서 백엔드 스냅샷의 이름을 지정합니다 annotations 현재 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
- `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`에서 ``snapshotHandle`` 지정합니다. 이 정보는 통화에서 외부 스냅샷이 Trident에 제공하는 유일한 ``ListSnapshots`` 정보입니다.



를 클릭합니다 `<volumeSnapshotContentName>` CR 명명 제한으로 인해 백엔드 스냅샷 이름과 항상 일치할 수 없습니다.

예

다음 예제에서는 을 만듭니다 VolumeSnapshotContent 백엔드 스냅샷을 참조하는 객체입니다 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: * 를 생성합니다 VolumeSnapshot 을 참조하는 CR VolumeSnapshotContent 오브젝트. 그러면 를 사용할 수 있는 액세스가 필요합니다 VolumeSnapshot 지정된 네임스페이스에서.

예

다음 예제에서는 을 만듭니다 VolumeSnapshot CR 이름 import-snap 을 참조합니다 VolumeSnapshotContent 이름 지정 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(조치 필요 없음): * 외부 스냅샷 작성자가 새로 생성된 을 VolumeSnapshotContent 인식하고 ListSnapshots 통화를 실행합니다. Trident가 를 `TridentSnapshot` 생성합니다.
 - 외부 스냅샷 작성기가 를 설정합니다 VolumeSnapshotContent 를 선택합니다 readyToUse 및 VolumeSnapshot 를 선택합니다 true.
 - Trident가 돌아왔습니다 readyToUse=true.
4. * 모든 사용자: * 를 생성합니다 PersistentVolumeClaim 를 눌러 새 를 참조합니다 VolumeSnapshot, 위치 spec.dataSource (또는 spec.dataSourceRef) name 은 입니다 VolumeSnapshot 이름.

예

다음 예에서는 를 참조하는 PVC를 작성합니다 VolumeSnapshot 이름 지정 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 데이터를 복구합니다

스냅샷 디렉토리는 를 사용하여 프로비저닝된 볼륨의 최대 호환성을 지원하기 위해 기본적으로 숨겨져 있습니다. ontap-nas 및 ontap-nas-economy 드라이버. 를 활성화합니다 .snapshot 스냅샷으로부터 직접 데이터를 복구할 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 대한 변경 사항은 손실됩니다.

스냅샷에서 전체 볼륨 복원

Trident는 (TASR) CR을 사용하여 스냅샷에서 제자리에서 신속하게 볼륨을 복원할 수 있도록 TridentActionSnapshotRestore 합니다. 이 CR은 필수 Kubernetes 조치로 작동하며 작업이 완료된 후에도 유지되지 않습니다.

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

시작하기 전에

바인딩된 PVC 및 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 Bound인지 확인한다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

1. TASR CR을 생성합니다. 이 예에서는 PVC 및 볼륨 스냅샷에 대한 CR을 pvc1 `pvc1-snapshot` 생성합니다.



TASR CR은 PVC & 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. CR을 적용하여 스냅샷에서 복원합니다. 이 예는 스냅샷에서 `pvc1` 복원합니다.

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

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

결과

Trident는 스냅샷에서 데이터를 복원합니다. 스냅샷 복구 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: pvcl
    volumeSnapshotName: pvcl-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""

```



- 대부분의 경우 Trident는 장애 발생 시 작업을 자동으로 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 권한이 없는 Kubernetes 사용자는 애플리케이션 네임스페이스에서 TASR CR을 생성할 수 있는 관리자의 권한을 받아야 할 수 있습니다.

연결된 스냅샷이 있는 PV를 삭제합니다

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

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포 시 스냅샷 컨트롤러와 CRD가 포함되지 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.yaml
1
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 경우를 엽니다 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 및 업데이트 namespace 네임스페이스로.

관련 링크

- ["볼륨 스냅샷"](#)
- ["VolumeSnapshotClass"](#)

볼륨 그룹 스냅샷 작업

영구 볼륨(PV)의 Kubernetes 볼륨 그룹 스냅샷 NetApp Trident는 여러 볼륨의 스냅샷(볼륨 스냅샷 그룹)을 생성할 수 있는 기능을 제공합니다. 이 볼륨 그룹 스냅샷은 동일 시점에 생성된 여러 볼륨의 복사본을 나타냅니다.



VolumeGroupSnapshot은 베타 API를 갖춘 쿠버네티스의 베타 기능입니다. VolumeGroupSnapshot을 사용하려면 쿠버네티스 1.32 버전이 필요합니다.

볼륨 그룹 스냅샷 생성

볼륨 그룹 스냅샷은 다음 스토리지 드라이버에서 지원됩니다.

- `ontap-san` 드라이버 - iSCSI 및 FC 프로토콜에만 적용되며 NVMe/TCP 프로토콜에는 적용되지 않습니다.
- `ontap-san-economy` - iSCSI 프로토콜에만 해당됩니다.
- 'ONTAP-NAS'



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

시작하기 전에

- Kubernetes 버전이 K8s 1.32 이상인지 확인하세요.
- 스냅샷을 사용하려면 외부 스냅샷 컨트롤러와 CRD(사용자 정의 리소스 정의)가 있어야 합니다. Kubernetes Orchestrator의 책임입니다(예: Kubeadm, GKE, OpenShift).

Kubernetes 배포판에 외부 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요. [볼륨 스냅샷 컨트롤러를 배포합니다](#).



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 생성합니다.
- VolumeGroupSnapshot을 생성하려면 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하세요.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하세요. 자세한 내용은 ["볼륨 그룹 스냅샷 클래스"](#) 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 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: pvcl-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 하나의 단위로 복구할 수는 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 전체 볼륨 복원

Trident는 (TASR) CR을 사용하여 스냅샷에서 제자리에서 신속하게 볼륨을 복원할 수 있도록 TridentActionSnapshotRestore 합니다. 이 CR은 필수 Kubernetes 조치로 작동하며 작업이 완료된 후에도 유지되지 않습니다.

자세한 내용은 을 ["스냅샷에서 전체 볼륨 복원"](#)참조하십시오.

연관된 그룹 스냅샷이 있는 PV 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- 그룹의 개별 스냅샷이 아닌 VolumeGroupSnapshots 전체를 삭제할 수 있습니다.
- 스냅샷이 있는 동안 해당 PersistentVolume이 삭제되면 Trident는 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 컨트롤러를 배포합니다

Kubernetes 배포 시 스냅샷 컨트롤러와 CRD가 포함되지 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_volumegroupsnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcontents.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 를 엽니다 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 및 업데이트 namespace 네임스페이스로.

관련 링크

- ["볼륨 그룹 스냅샷 클래스"](#)
- ["볼륨 스냅샷"](#)

저작권 정보

Copyright © 2026 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이선스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이선스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 <http://www.netapp.com/TM>에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.