볼륨 작업을 수행합니다

Astra Trident

NetApp
April 16, 2024

목차

볼륨 작업을 수행합니다

볼륨 관리를 위한 Astra Trident의 기능에 대해 알아보십시오.

- "CSI 토폴로지를 사용합니다"
- "스냅샷 작업"
- "볼륨 확장"
- "볼륨 가져오기"

CSI 토폴로지를 사용합니다

Astra Trident는 을 사용하여 Kubernetes 클러스터에 있는 노드를 선택적으로 생성하여 연결할 수 있습니다 "CSI 토폴로지 기능". 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. Astra Trident는 다중 영역 아키텍처에서 워크로드용 볼륨 프로비저닝을 지원하기 위해 CSI 토폴로지를 사용합니다.

CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

- 와 함께 `VolumeBindingMode` 를 로 설정합니다 `Immediate`, Astra Trident는 토폴로지 인식 없이 볼륨을 생성합니다. 볼륨 바인딩 및 동적 프로비저닝은 PVC가 생성될 때 처리됩니다. 이것이 기본값입니다 `VolumeBindingMode` 또한 토폴로지 제약 조건을 적용하지 않는 클러스터에 적합합니다. 영구 볼륨은 요청 포드의 예약 요구 사항에 의존하지 않고 생성됩니다.
- 와 함께 `VolumeBindingMode` 를 로 설정합니다 `WaitForFirstConsumer`PVC에 대한 영구 볼륨의 생성 및 바인딩은 PVC를 사용하는 POD가 예약 및 생성될 때까지 지연됩니다. 이렇게 하면 토폴로지 요구 사항에 따라 적용되는 일정 제한을 충족하기 위해 볼륨이 생성됩니다.

를 클릭합니다 `WaitForFirstConsumer` 바인딩 모드에서는 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:"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 토폴로지를 인식하는 레이블이 있어야 합니다 (`topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`)를 클릭합니다. Astra Trident가 토폴로지 인식을 위해 설치되기 전에 클러스터의 노드에 이러한 레이블 * 이 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

Astra Trident 스토리지 백엔드는 가용성 영역에 따라 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드에는 선택 사항이 포함될 수 있습니다 `supportedTopologies` 지원해야 하는 영역 및 영역 목록을 나타내는 블록입니다. 이러한 백엔드를 사용하는 StorageClasses의 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` 백엔드 당 지역 및 영역 목록을 제공하는 데 사용됩니다. 이러한 영역 및 영역은 StorageClass 에서 제공할 수 있는 허용 가능한 값의 목록을 나타냅니다. 백엔드에서 제공되는 지역 및 영역의 하위 집합이 포함된 StorageClasses의 경우 Astra Trident는 백엔드에 볼륨을 생성합니다.

을 정의할 수 있습니다 `supportedTopologies` 스토리지 풀당 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
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
```

이 예에서 는 입니다 `region` 및 `zone` 레이블은 스토리지 풀의 위치를 나타냅니다. `topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone` 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. 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"
```

위에서 제공한 StorageClass 정의에서 `volumeBindingMode` 가 로 설정되어 있습니다 `WaitForFirstConsumer`. 이 StorageClass에 요청된 PVC는 POD에서 참조될 때까지 작동하지 않습니다. 그리고, `allowedTopologies` 사용할 영역 및 영역을 제공합니다. 를 클릭합니다 `netapp-san-us-east1` StorageClass가 에 PVC를 생성합니다 `san-backend-us-east1` 백엔드가 위에서 정의되었습니다.

3단계: PVC 생성 및 사용

StorageClass가 생성되어 백엔드에 매핑되면 PVC를 생성할 수 있습니다.

예를 참조하십시오 `spec` 아래:

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

이 매니페스트를 사용하여 PVC를 만들면 다음과 같은 결과가 발생합니다.

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

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: 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 에 있는 노드에서 Pod를 예약하도록 Kubernetes에 지시합니다 `us-east1` 영역 을 클릭하고 에 있는 노드 중에서 선택합니다 `us-east1-a` 또는 `us-east1-b` 존.

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 목록을 포함하도록 업데이트할 수 있습니다 `supportedTopologies` 사용 `tridentctl backend update`. 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

PVS(Persistent Volumes)의 Kubernetes VolumeSnapshots(볼륨 스냅샷)을 생성하여 Astra Trident 볼륨의 시점 복제본을 유지할 수 있습니다. 또한 기존 볼륨 스냅샷에서 _clone_이라고도 하는 새 볼륨을 생성할 수 있습니다. 에서 볼륨 스냅샷을 지원합니다 `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, 및 `azure-netapp-files` 드라이버.

시작하기 전에

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

Kubernetes 배포 시 스냅샷 컨트롤러 및 CRD가 포함되지 않은 경우 를 참조하십시오 볼륨 스냅샷 컨트롤러 배포.

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

1단계: 을 작성합니다 `VolumeSnapshotClass`

이 예에서는 볼륨 스냅샷 클래스를 생성합니다.

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

를 클릭합니다 `driver` Astra Trident의 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.

자세한 내용은 LINK:../trident-reference/objects.html#Kubernetes-volumesshotshotclass-objects를 참조하십시오 [`VolumeSnapshotClass`].

2단계: 기존 PVC의 스냅샷을 생성합니다

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 예제에서는 이름이 인 PVC에 대해 스냅샷이 생성됩니다 `pvc1` 스냅샷 이름이 로 설정되어 있습니다 `pvc1-snap`.

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

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

이렇게 하면 가 생성됩니다 `VolumeSnapshot` 오브젝트. VolumeSnapshot은 PVC와 유사하며 와 관련이 있습니다 `VolumeSnapshotContent` 실제 스냅샷을 나타내는 객체입니다.

를 식별할 수 있습니다 `VolumeSnapshotContent` 의 개체 `pvc1-snap` VolumeSnapshot을 설명합니다.

```
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
.
.
```

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.

3단계: VolumeSnapshots에서 PVC를 생성합니다

다음은 스냅샷을 사용하여 PVC를 생성하는 예제입니다.

```
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
```

`dataSource` 는 이름이 인 VolumeSnapshot을 사용하여 PVC를 생성해야 함을 나타냅니다 `pvc1-snap` 데이터 소스로 사용됩니다. 이렇게 하면 Astra Trident가 스냅샷에서 PVC를 생성하도록 지시합니다. PVC가 생성된 후 POD에 부착하여 다른 PVC와 마찬가지로 사용할 수 있습니다.

> 연결된 스냅샷이 있는 영구 볼륨을 삭제하면 해당 Trident 볼륨이 "삭제 상태"로 업데이트됩니다. Astra Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

볼륨 스냅샷 컨트롤러 배포

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.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. 원하는 네임스페이스에 스냅샷 컨트롤러를 생성합니다. 아래 YAML 매니페스트를 편집하여 네임스페이스를 수정하십시오.

```
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
```

관련 링크

- "볼륨 스냅숏"
- "VolumeSnapshotClass"

볼륨 확장

Astra Trident를 사용하면 Kubernetes 사용자가 볼륨을 생성한 후 확장할 수 있습니다. iSCSI 및 NFS 볼륨을 확장하는 데 필요한 구성에 대한 정보를 찾습니다.

iSCSI 볼륨을 확장합니다

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

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

개요

iSCSI PV를 확장하면 다음 단계가 포함됩니다.

- StorageClass 정의를 편집하여 를 설정합니다 `allowVolumeExpansion` 필드를 눌러 로 이동합니다 `true`.
- PVC 정의 편집 및 업데이트 `spec.resources.requests.storage` 원래 크기보다 커야 하는 새로 원하는 크기를 반영합니다.
- 크기를 조절하려면 PV를 포드에 부착해야 합니다. iSCSI PV의 크기를 조정할 때 두 가지 시나리오가 있습니다.
  - PV가 포드에 연결된 경우 Astra Trident는 스토리지 백엔드의 볼륨을 확장하고 디바이스를 다시 검사하며 파일 시스템의 크기를 조정합니다.
  - 연결되지 않은 PV의 크기를 조정하려고 하면 Astra Trident가 스토리지 백엔드의 볼륨을 확장합니다. PVC가 POD에 바인딩되면 Trident가 디바이스를 다시 검사해 파일 시스템의 크기를 조정합니다. 그런 다음 확장 작업이 성공적으로 완료된 후 Kubernetes에서 PVC 크기를 업데이트합니다.

아래 예에서는 iSCSI PVS의 확장 작동 방식을 보여 줍니다.

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

```
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 StorageClass 의 경우 를 포함하도록 편집합니다 `allowVolumeExpansion` 매개 변수.

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

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

Astra Trident가 PV(Persistent Volume)를 생성하여 이 PVC(Persistent Volume Claim)와 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

이 예제에서는 을 사용하는 포드가 만들어집니다 `san-pvc`.

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

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

4단계: **PV**를 확장합니다

1Gi 에서 2Gi 로 생성된 PV 의 크기를 조정하려면 PVC 정의를 편집하고 를 업데이트합니다 `spec.resources.requests.storage` 2Gi

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

5단계**:** 확장 확인

PVC, PV, Astra Trident 볼륨의 크기를 확인하여 확장이 제대로 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 볼륨을 확장합니다

Astra Trident는 에 프로비저닝된 NFS PVS에 대한 볼륨 확장을 지원합니다 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `gcp-cvs`, 및 `azure-netapp-files` 백엔드.

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

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

```
cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

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

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
```

Astra 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**를 확장합니다

새로 생성된 20MiB PV의 크기를 1GiB로 조정하려면 PVC를 편집하고 설정합니다
`spec.resources.requests.storage` 1GB:

```
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, Astra Trident 볼륨의 크기를 확인하여 크기가 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

볼륨 가져오기

를 사용하여 기존 스토리지 볼륨을 Kubernetes PV로 가져올 수 있습니다 `tridentctl import`.

볼륨 가져오기를 지원하는 드라이버입니다

이 표에는 볼륨 가져오기를 지원하는 드라이버와 볼륨 가져오기가 도입된 릴리스가 나와 있습니다.

드라이버	놓습니다
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

드라이버	놓습니다
gcp-cvs	19.04
ontap-san	19.04

볼륨을 가져와야 하는 이유는 무엇입니까?

Trident로 볼륨을 가져오는 데는 다음과 같은 몇 가지 사용 사례가 있습니다.

- 응용 프로그램을 Containerizing 하고 기존 데이터 집합을 다시 사용합니다
- 수명이 짧은 애플리케이션에 대한 데이터 세트 클론 사용
- 오류가 발생한 Kubernetes 클러스터를 리빌드합니다
- 재해 복구 중에 애플리케이션 데이터 마이그레이션

가져오기는 어떻게 작동합니까?

영구 볼륨 클레임(PVC) 파일은 볼륨 가져오기 프로세스에서 PVC를 생성하는 데 사용됩니다. 최소한 PVC 파일에는 다음 예제와 같이 이름, 네임스페이스, accessModes 및 storageClassName 필드가 포함되어야 합니다.

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

를 클릭합니다 `tridentctl` 클라이언트는 기존 스토리지 볼륨을 가져오는 데 사용됩니다. Trident는 볼륨 메타데이터를 유지하고 PVC 및 PV를 생성하여 볼륨을 가져옵니다.

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

스토리지 볼륨을 가져오려면 볼륨을 포함하는 Astra Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름을 지정합니다(예: ONTAP FlexVol, Element Volume, CVS 볼륨 경로). 스토리지 볼륨은 읽기/쓰기 액세스를 허용해야 하며 지정된 Astra Trident 백엔드에서 액세스할 수 있어야 합니다. 를 클릭합니다 `-f` 문자열 인수가 필요하며 YAML 또는 JSON PVC 파일의 경로를 지정합니다.

Astra Trident가 볼륨 가져오기 요청을 받으면 기존 볼륨 크기를 결정하고 PVC에 설정합니다. 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.

를 사용하여 볼륨을 가져올 때 `--no-manage` argument, Trident는 개체의 수명 주기 동안 PVC 또는 PV에 대한 추가 작업을 수행하지 않습니다. Trident는 에 대한 PV 및 PVC 이벤트를 무시하므로 `--no-manage` 객체, PV 삭제 시 스토리지 볼륨이 삭제되지 않습니다. 볼륨 클론 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다. 이 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하고, 그렇지 않고 Kubernetes 외부 스토리지 볼륨의 라이프사이클을 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.

Trident 19.07 이상 버전에서는 PVS 연결을 처리하고 볼륨을 가져오는 과정에서 볼륨을 마운트합니다. 이전 버전의 Astra Trident를 사용하여 가져오는 경우 데이터 경로에 작업이 없으며 볼륨 가져오기에서 볼륨을 마운트할 수 있는지 여부를 확인하지 않습니다. 볼륨 가져오기에서 오류가 발생한 경우(예: StorageClass가 올바르지 않은 경우) PV에 대한 부가세 반환 청구액 정책을 로 변경하여 복구할 수 있습니다 `retain`PVC와 PV를 삭제하고 볼륨 가져오기 명령을 다시 시도합니다.

`ontap-nas` 및 `ontap-nas-flexgroup` 가져오기

로 생성된 각 볼륨입니다 `ontap-nas` 드라이버는 ONTAP 클러스터의 FlexVol입니다. 를 사용하여 FlexVol을 가져옵니다 `ontap-nas` 드라이버는 동일하게 작동합니다. ONTAP 클러스터에 이미 있는 FlexVol를 로 가져올 수 있습니다 `ontap-nas` PVC. 마찬가지로 FlexGroup vols는 로 가져올 수 있습니다 `ontap-nas-flexgroup` PVC

ONTAP 볼륨을 Trident에서 가져오려면 RW 유형이어야 합니다. 볼륨이 DP 유형인 경우 SnapMirror 대상 볼륨이므로 볼륨을 Trident로 가져오기 전에 미러 관계를 끊어야 합니다.

를 클릭합니다 `ontap-nas` 드라이버가 qtree를 가져오고 관리할 수 없습니다. 를 클릭합니다 `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버는 중복 볼륨 이름을 허용하지 않습니다.

예를 들어, 이라는 이름의 볼륨을 가져옵니다 `managed_volume` 백엔드에서 을(를) 선택합니다 `ontap_nas`에서 다음 명령을 사용합니다.

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

을 눌러 라는 이름의 볼륨을 가져옵니다 `unmanaged_volume` (에서 `ontap_nas backend`), Trident에서 관리하지 않을 경우 다음 명령을 사용합니다.

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

를 사용할 때 `--no-manage` argument, Trident는 볼륨의 이름을 바꾸거나 볼륨이 마운트되었는지 검증하지 않습니다. 볼륨이 수동으로 마운트되지 않은 경우 볼륨 가져오기 작업이 실패합니다.

사용자 지정 UnixPermissions를 사용하여 볼륨을 가져오는 기존 버그가 수정되었습니다. PVC 정의 또는 백엔드 구성에서 unixPermissions를 지정하고 Astra Trident에 볼륨을 가져오도록 지시할 수 있습니다.

`ontap-san` 가져오기

Astra Trident는 단일 LUN이 포함된 ONTAP SAN FlexVol도 가져올 수 있습니다. 이는 와 일치합니다 `ontap-san` 드라이버 - 각 PVC 및 FlexVol 내의 LUN에 대한 FlexVol를 생성합니다. 를 사용할 수 있습니다 `tridentctl import` 다른 경우와 동일한 방식으로 명령을 실행합니다.

- 의 이름을 포함합니다 `ontap-san` 백엔드.
- 가져올 FlexVol의 이름을 입력합니다. 이 FlexVol에는 가져와야 하는 LUN이 하나만 포함되어 있습니다.
- 와 함께 사용해야 하는 PVC 정의의 경로를 제공합니다 `-f` 깃발.
- PVC 관리 또는 비관리형 중에서 선택합니다. 기본적으로 Trident는 PVC를 관리하고 백엔드에서 FlexVol 및 LUN의 이름을 바꿉니다. 관리되지 않는 볼륨으로 가져오려면 를 전달합니다 `--no-manage` 깃발.

관리되지 않는 파일을 가져올 때 `ontap-san` 볼륨, FlexVol의 LUN 이름이 지정되었는지 확인해야 합니다 `lun0` 원하는 이니시에이터가 있는 igroup에 매핑되어 있습니다. Astra Trident에서 관리되는 가져오기를 위해 이 작업을 자동으로 처리합니다.

그러면 Astra Trident가 FlexVol를 가져와 PVC 정의와 연결합니다. Astra Trident도 FlexVol의 이름을 로 바꿉니다 `pvc-<uuid>` 및 FlexVol 내의 LUN을 에 포맷합니다 `lun0`.

기존 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 사용 중인 볼륨을 가져오려는 경우 먼저 볼륨을 클론한 다음 가져오기를 수행합니다.

예

을(를) 가져옵니다 `ontap-san-managed` 에 있는 FlexVol입니다 `ontap_san_default` 백엔드에서 를 실행합니다 `tridentctl import` 명령:

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

Astra Trident에서 가져오려면 ONTAP 볼륨이 RW 유형이어야 합니다. 볼륨이 DP 유형인 경우 SnapMirror 대상 볼륨이므로 볼륨을 Astra Trident로 가져오기 전에 미러 관계를 끊어야 합니다.

`element` 가져오기

Trident를 사용하여 NetApp Element 소프트웨어/NetApp HCI 볼륨을 Kubernetes 클러스터로 가져올 수 있습니다. Astra Trident 백엔드의 이름과 볼륨의 고유 이름 및 PVC 파일이 의 인수로 필요합니다 `tridentctl import` 명령.

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

Element 드라이버는 중복 볼륨 이름을 지원합니다. 중복된 볼륨 이름이 있는 경우 Trident의 볼륨 가져오기 프로세스에서 오류가 반환됩니다. 이 문제를 해결하려면 볼륨을 복제하여 고유한 볼륨 이름을 제공합니다. 그런 다음 복제된 볼륨을 가져옵니다.

gcp-cvs 가져오기

GCP에서 NetApp Cloud Volumes Service가 지원하는 볼륨을 가져오려면 해당 이름 대신 볼륨 경로를 기준으로 볼륨을 식별합니다.

을(를) 가져옵니다 gcp-cvs 백엔드의 볼륨을 호출했습니다 gcpcvs_YEppr 볼륨 경로 포함 `adroit-jolly-swift`에서 다음 명령을 사용합니다.

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

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

볼륨 경로는 / 이후의 볼륨 내보내기 경로 부분입니다. 예를 들어, 내보내기 경로가 인 경우 10.0.0.1:/adroit-jolly-swift, 볼륨 경로는 입니다 adroit-jolly-swift.

azure-netapp-files 가져오기

을(를) 가져옵니다 azure-netapp-files 백엔드의 볼륨을 호출했습니다 azurenetappfiles_40517 볼륨 경로 포함 `importvol1`에서 다음 명령을 실행합니다.

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

ANF 볼륨의 볼륨 경로는 다음:/ 이후의 마운트 경로에 있습니다. 예를 들어, 마운트 경로가 인 경우 `10.0.0.2:/importvol1`, 볼륨 경로는 입니다 `importvol1`.