Trident 25.06 설명서

Trident

NetApp
January 15, 2026

목차

Trident 25.06 설명서

릴리스 노트

새로운 소식

릴리스 노트는 NetApp Trident 최신 버전의 새로운 기능, 향상된 기능 및 버그 수정에 대한 정보를 제공합니다.

> 그만큼 `tridentctl` 설치 프로그램 zip 파일에 제공된 Linux용 바이너리는 테스트를 거쳐 지원되는 버전입니다. 다음 사항을 주의하세요. `macos` 에서 제공되는 바이너리 `/extras` zip 파일의 일부는 테스트되지 않았거나 지원되지 않습니다.

25.06.2의 새로운 기능

새로운 기능 요약에서는 Trident 와 Trident Protect 릴리스에 대한 개선 사항, 수정 사항 및 사용 중단에 대한 세부 정보를 제공합니다.

Trident

수정 사항

- **Kubernetes**: Kubernetes 노드에서 볼륨을 분리할 때 잘못된 iSCSI 장치가 검색되는 심각한 문제가 해결되었습니다.

25.06.1의 변경 사항

Trident

> SolidFire 사용하는 고객의 경우 볼륨 게시 취소 시 발생하는 알려진 문제로 인해 25.06.1로 업그레이드하지 마십시오. 이 문제를 해결하기 위해 25.06.2가 곧 출시될 예정입니다.

수정 사항

- 쿠버네티스:
  - 하위 시스템에서 매핑 해제되기 전에 NQN이 확인되지 않는 문제가 해결되었습니다.
  - LUKS 장치를 닫으려고 여러 번 시도하면 볼륨 분리에 실패하는 문제가 해결되었습니다.
  - 장치 경로가 생성된 이후 변경된 경우 고정 iSCSI 볼륨이 스테이지 해제됩니다.
  - 스토리지 클래스 전반에 걸쳐 볼륨을 블록 복제합니다.
- **OpenShift**: OCP 4.19에서 iSCSI 노드 준비가 실패하는 문제를 해결했습니다.
- SolidFire 백엔드를 사용하여 볼륨을 복제할 때 시간 초과가 증가했습니다."1008호").

25.06의 변경 사항

Trident

개선 사항

- 쿠버네티스:
  - CSI 볼륨 그룹 스냅샷에 대한 지원이 추가되었습니다. `v1beta1` ONTAP-SAN iSCSI 드라이버를 위한 볼륨 그룹 스냅샷 Kubernetes API입니다. 보다"볼륨 그룹 스냅샷 작업" .

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

  - iSCSI 외에도 NVMe/TCP용 ONTAP ASA r2에 대한 지원이 추가되었습니다. 보다link:"ONTAP SAN 구성 옵션 및 예" .
  - ONTAP-NAS 및 ONTAP-NAS-Economy 볼륨에 대한 보안 SMB 지원이 추가되었습니다. 이제 Active Directory 사용자와 그룹을 SMB 볼륨과 함께 사용하여 보안을 강화할 수 있습니다. 보다"보안 SMB 활성화" .
  - iSCSI 볼륨의 노드 작업에서 더 높은 확장성을 제공하기 위해 향상된 Trident 노드 동시성을 제공합니다.
  - 추가됨 `--allow-discards` LUKS 볼륨을 열 때 공간 회수를 위한 삭제/TRIM 명령을 허용합니다.
  - LUKS로 암호화된 볼륨을 포맷할 때 성능이 향상되었습니다.
  - 오류가 발생했지만 부분적으로 포맷된 LUKS 장치에 대한 향상된 LUKS 정리 기능입니다.
  - NVMe 볼륨 연결 및 분리를 위한 향상된 Trident 노드 멱등성.
  - 추가됨 `internalID` ONTAP -SAN-Economy 드라이버에 대한 Trident 볼륨 구성에 대한 필드입니다.
  - NVMe 백엔드에 SnapMirror 사용한 볼륨 복제 지원이 추가되었습니다. 보다"SnapMirror 사용하여 볼륨 복제" .

실험적 개선 사항

프로덕션 환경에서는 사용할 수 없습니다.

- [기술 미리보기] 다음을 통해 동시 Trident 컨트롤러 작업이 활성화되었습니다. `--enable-concurrency` 기능 플래그. 이를 통해 컨트롤러 작업을 병렬로 실행할 수 있어 작업량이 많거나 대규모 환경에서 성능이 향상됩니다.

  이 기능은 실험적이며 현재 ONTAP-SAN 드라이버(iSCSI 및 FCP 프로토콜)를 사용하여 제한된 병렬 워크플로를 지원합니다.

- [기술 미리보기] ANF 드라이버에 수동 QOS 지원이 추가되었습니다.

수정 사항

- 쿠버네티스:
  - 기본 SCSI 디스크를 사용할 수 없는 경우 멀티패스 장치의 크기가 일치하지 않는 CSI NodeExpandVolume 문제가 해결되었습니다.
  - ONTAP-NAS 및 ONTAP-NAS-Economy 드라이버에 대한 중복된 내보내기 정책을 정리하지 못하는 문제가 수정되었습니다.
  - 고정된 GCNV 볼륨이 NFSv3로 기본 설정됨 `nfsMountOptions` 설정되지 않았습니다. 이제 NFSv3 및 NFSv4 프로토콜이 모두 지원됩니다. 만약에 `nfsMountOptions` 제공되지 않으면 호스트의 기본 NFS 버전(NFSv3 또는 NFSv4)이 사용됩니다.

- Kustomize를 사용하여 Trident 설치할 때 고정된 배포 문제("831호").
- 스냅샷에서 생성된 PVC에 대한 누락된 내보내기 정책이 수정되었습니다."1016호").
- ANF 볼륨 크기가 1GiB 단위로 자동 정렬되지 않는 문제가 해결되었습니다.
- Bottlerocket과 함께 NFSv3를 사용할 때 발생하는 문제가 해결되었습니다.

- SolidFire 백엔드를 사용하여 볼륨을 복제할 때 고정된 시간 초과("1008호").
- 크기 조정에 실패했음에도 불구하고 ONTAP-NAS-Economy 볼륨이 최대 300TB까지 확장되는 문제가 해결되었습니다.
- ONTAP REST API를 사용할 때 복제 분할 작업이 동기적으로 수행되는 문제가 해결되었습니다.

사용 중단:

- **Kubernetes**: 지원되는 최소 Kubernetes 버전이 v1.27로 업데이트되었습니다.

Trident 프로텍트

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너가 지원하는 상태 저장 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다.

개선 사항

- 더 자주 전체 백업을 수행할 수 있는 옵션을 제공하여 복구 시간이 단축되었습니다.
- 그룹-버전-종류(GVK) 필터링을 통해 애플리케이션 정의의 세분성이 향상되고 선택적 복원이 가능해졌습니다.
- NetApp SnapMirror 와 함께 AppMirrorRelationship(AMR)을 사용하면 전체 PVC 복제를 방지하기 위해 효율적인 재동기화 및 역방향 복제가 가능합니다.
- EKS Pod Identity를 사용하여 AppVault 버킷을 생성하는 기능이 추가되어 EKS 클러스터에 대한 버킷 자격 증명으로 비밀을 지정할 필요가 없어졌습니다.
- 필요한 경우 복원 네임스페이스에서 라벨과 주석 복원을 건너뛸 수 있는 기능이 추가되었습니다.
- AppMirrorRelationship(AMR)은 이제 소스 PVC 확장을 확인하고 필요에 따라 대상 PVC에서 적절한 확장을 수행합니다.

수정 사항

- 이전 스냅샷의 스냅샷 주석 값이 최신 스냅샷에 적용되는 버그를 수정했습니다. 이제 모든 스냅샷 주석이 올바르게 적용됩니다.
- 기본적으로 데이터 이동자 암호화(Kopia/Restic)를 위한 비밀이 정의되어 있습니다(정의되지 않은 경우).
- S3 AppVault 생성에 대한 검증 및 오류 메시지가 개선되었습니다.
- AppMirrorRelationship(AMR)은 이제 실패한 시도를 방지하기 위해 Bound 상태의 PV만 복제합니다.
- 대량의 백업이 있는 AppVault에서 AppVaultContent를 가져올 때 오류가 표시되는 문제가 해결되었습니다.
- KubeVirt VMSnapshot은 장애를 방지하기 위해 복원 및 장애 조치 작업에서 제외됩니다.
- Kopia의 기본 보존 일정이 사용자가 일정에 설정한 내용을 무시하고, 이로 인해 스냅샷이 조기에 제거되는 문제가 해결되었습니다.

25.02.1의 변경 사항

Trident

수정 사항

- 쿠버네티스:
  - 기본이 아닌 이미지 레지스트리를 사용할 때 사이드카 이미지 이름과 버전이 잘못 채워지는 trident-operator의 문제를 해결했습니다.("983호").
  - ONTAP 장애 조치(Giveback) 중에 다중 경로 세션이 복구되지 않는 문제를 해결했습니다.("961호").

25.02의 변경 사항

Trident 25.02부터 새로운 기능 요약에서는 Trident 와 Trident Protect 릴리스에 대한 개선 사항, 수정 사항 및 사용 중단에 대한 세부 정보를 제공합니다.

Trident

개선 사항

- 쿠버네티스:
  - iSCSI에 대한 ONTAP ASA r2 지원이 추가되었습니다.
  - 비정상적인 노드 종료 시나리오에서 ONTAP-NAS 볼륨에 대한 강제 분리 지원이 추가되었습니다. 새로운 ONTAP-NAS 볼륨은 이제 Trident 가 관리하는 볼륨별 내보내기 정책을 활용합니다. 활성 작업 부하에 영향을 주지 않고 게시 취소 시 기존 볼륨이 새로운 내보내기 정책 모델로 전환될 수 있도록 업그레이드 경로를 제공했습니다.
  - cloneFromSnapshot 주석을 추가했습니다.
  - 네임스페이스 간 볼륨 복제에 대한 지원이 추가되었습니다.
  - 정확한 호스트, 채널, 대상 및 LUN ID로 재스캔을 시작하기 위해 향상된 iSCSI 자체 복구 스캔 수정 기능이 추가되었습니다.
  - Kubernetes 1.32에 대한 지원이 추가되었습니다.
- 오픈시프트:
  - ROSA 클러스터에서 RHCOS에 대한 자동 iSCSI 노드 준비 지원이 추가되었습니다.
  - ONTAP 드라이버에 대한 OpenShift Virtualization 지원이 추가되었습니다.
- ONTAP-SAN 드라이버에 파이버 채널 지원이 추가되었습니다.
- NVMe LUKS 지원이 추가되었습니다.
- 모든 기본 이미지에 대해 스크래치 이미지로 전환했습니다.
- iSCSI 세션이 로그인되어야 하지만 로그인되지 않은 경우 iSCSI 연결 상태 검색 및 로깅이 추가되었습니다.("961호").
- google-cloud-netapp-volumes 드라이버를 사용하여 SMB 볼륨에 대한 지원이 추가되었습니다.
- 삭제 시 ONTAP 볼륨이 복구 대기열을 건너뛸 수 있도록 지원이 추가되었습니다.
- 태그 대신 SHA를 사용하여 기본 이미지를 재정의하는 지원이 추가되었습니다.

- tridentctl 설치 프로그램에 image-pull-secrets 플래그를 추가했습니다.

수정 사항

- 쿠버네티스:
  - 자동 내보내기 정책에서 누락된 노드 IP 주소가 수정되었습니다("965호").
  - ONTAP-NAS-Economy의 경우 자동 내보내기 정책이 볼륨별 정책으로 조기에 전환되는 문제가 수정되었습니다.
  - 사용 가능한 모든 AWS ARN 파티션을 지원하기 위해 고정 백엔드 구성 자격 증명("913호").
  - Trident 운영자에서 자동 구성 조정을 비활성화하는 옵션이 추가되었습니다."924호").
  - csi-resizer 컨테이너에 대한 securityContext가 추가되었습니다."976호").

Trident 프로텍트

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너가 지원하는 상태 저장 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다.

개선 사항

- volumeMode: File 및 volumeMode: Block(원시 장치) 스토리지 모두에 대해 KubeVirt/OpenShift 가상화 VM에 대한 백업 및 복원 지원이 추가되었습니다. 이 지원은 모든 Trident 드라이버와 호환되며, Trident Protect와 함께 NetApp SnapMirror 사용하여 스토리지를 복제할 때 기존 보호 기능을 향상시킵니다.
- Kubevirt 환경에서 애플리케이션 수준에서 동결 동작을 제어하는 기능이 추가되었습니다.
- AutoSupport 프록시 연결 구성에 대한 지원이 추가되었습니다.
- 데이터 이동자 암호화(Kopia/Restic)에 대한 비밀을 정의하는 기능이 추가되었습니다.
- 수동으로 실행 후크를 실행할 수 있는 기능이 추가되었습니다.
- Trident Protect 설치 중에 보안 컨텍스트 제약 조건(SCC)을 구성하는 기능이 추가되었습니다.
- Trident Protect 설치 중 nodeSelector 구성에 대한 지원이 추가되었습니다.
- AppVault 개체에 대한 HTTP/HTTPS 송신 프록시 지원이 추가되었습니다.
- 클러스터 범위 리소스를 제외할 수 있도록 ResourceFilter를 확장했습니다.
- S3 AppVault 자격 증명에 AWS 세션 토큰에 대한 지원이 추가되었습니다.
- 스냅샷 실행 후크 이후 리소스 수집에 대한 지원이 추가되었습니다.

수정 사항

- ONTAP 볼륨 복구 대기열을 건너뛰기 위해 임시 볼륨 관리가 개선되었습니다.
- SCC 주석이 이제 원래 값으로 복원되었습니다.
- 병렬 작업 지원을 통해 복구 효율성이 향상되었습니다.
- 대규모 애플리케이션에 대한 실행 후크 타임아웃에 대한 지원이 향상되었습니다.

24.10.1의 변경 사항

개선 사항

- **Kubernetes**: Kubernetes 1.32에 대한 지원이 추가되었습니다.
- iSCSI 세션이 로그인되어야 하지만 로그인되지 않은 경우 iSCSI 연결 상태 검색 및 로깅이 추가되었습니다."961호").

수정 사항

- 자동 내보내기 정책에서 누락된 노드 IP 주소가 수정되었습니다("965호").
- ONTAP-NAS-Economy의 경우 자동 내보내기 정책이 볼륨별 정책으로 조기에 전환되는 문제가 수정되었습니다.
- CVE-2024-45337 및 CVE-2024-45310을 해결하기 위해 Trident 및 Trident-ASUP 종속성을 업데이트했습니다.
- iSCSI 자체 복구 중 간헐적으로 비정상인 비CHAP 포털에 대한 로그아웃이 제거되었습니다."961호").

24.10의 변경 사항

개선 사항

- Google Cloud NetApp Volumes 드라이버는 이제 NFS 볼륨에 대해 일반적으로 사용 가능하며 영역 인식 프로비저닝을 지원합니다.
- GCP 워크로드 ID는 GKE를 사용하는 Google Cloud NetApp Volumes 의 클라우드 ID로 사용됩니다.
- 추가됨 `formatOptions` 사용자가 LUN 형식 옵션을 지정할 수 있도록 ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버에 대한 구성 매개변수입니다.
- Azure NetApp Files 최소 볼륨 크기를 50GiB로 줄였습니다. Azure의 새로운 최소 크기는 11월에 일반적으로 출시될 예정입니다.
- 추가됨 `denyNewVolumePools` ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버를 기존 Flexvol 풀로 제한하는 구성 매개변수입니다.
- 모든 ONTAP 드라이버에서 SVM의 집계 추가, 제거 또는 이름 변경에 대한 감지 기능이 추가되었습니다.
- 보고된 PVC 크기를 사용할 수 있도록 LUKS LUN에 18MiB 오버헤드를 추가했습니다.
- ONTAP-SAN 및 ONTAP-SAN-Economy 노드의 스테이지 및 언스테이지 오류 처리가 개선되어 스테이지 실패 후 언스테이지에서 장치를 제거할 수 있습니다.
- ONTAP 에서 Trident 에 대한 최소한의 역할을 고객이 생성할 수 있도록 사용자 정의 역할 생성기를 추가했습니다.
- 문제 해결을 위한 추가 로깅이 추가되었습니다. `lsscsi` ("792호").

쿠버네티스

- Kubernetes 기반 워크플로를 위한 새로운 Trident 기능이 추가되었습니다.
  - 데이터 보호
  - 데이터 마이그레이션
  - 재해 복구
  - 애플리케이션 모빌리티

"Trident Protect에 대해 자세히 알아보세요".

- 새로운 플래그를 추가했습니다 `--k8s-api-qps` Trident 가 Kubernetes API 서버와 통신하는 데 사용하는 QPS 값을 설정하도록 설치자에게 요청합니다.
- 추가됨 `--node-prep` Kubernetes 클러스터 노드에서 스토리지 프로토콜 종속성을 자동으로 관리하기 위해 설치 프로그램에 플래그를 지정합니다. Amazon Linux 2023 iSCSI 스토리지 프로토콜과의 호환성이 테스트 및 검증되었습니다.
- Non-Graceful Node Shutdown 시나리오 동안 ONTAP-NAS-Economy 볼륨에 대한 강제 분리 지원이 추가되었습니다.
- 새로운 ONTAP-NAS-Economy NFS 볼륨은 다음을 사용할 때 Qtree당 내보내기 정책을 사용합니다. `autoExportPolicy` 백엔드 옵션. Qtree는 액세스 제어 및 보안을 개선하기 위해 게시 시점에 노드 제한적 내보내기 정책에만 매핑됩니다. Trident 활성 작업 부하에 영향을 주지 않고 모든 노드에서 볼륨의 게시를 취소하면 기존 qtree는 새로운 내보내기 정책 모델로 전환됩니다.
- Kubernetes 1.31에 대한 지원이 추가되었습니다.

실험적 개선 사항

- ONTAP-SAN 드라이버에 대한 파이버 채널 지원에 대한 기술 미리보기가 추가되었습니다.

수정 사항

- 쿠버네티스:
  - Trident Helm 설치를 방해하는 고정 Rancher 입장 웹훅("839호").
  - 헬름 차트 값의 고정된 Affinity 키("898호").
  - 고정된 tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector가 "true" 값으로 작동하지 않습니다."899호").
  - 복제 중에 생성된 임시 스냅샷을 삭제했습니다."901호").
- Windows Server 2019에 대한 지원이 추가되었습니다.
- Trident repo에서 `go mod tidy`를 수정했습니다."767호").

사용 중단

- 쿠버네티스**:**
  - 지원되는 최소 Kubernetes 버전이 1.25로 업데이트되었습니다.
  - POD 보안 정책에 대한 지원이 제거되었습니다.

제품 리브랜딩

24.10 릴리스부터 Astra Trident Trident (Netapp Trident)로 리브랜딩되었습니다. 이 리브랜딩은 Trident 의 기능, 지원 플랫폼 또는 상호 운용성에는 영향을 미치지 않습니다.

24.06의 변경 사항

개선 사항

- 중요: `limitVolumeSize` 매개변수는 이제 ONTAP Economy 드라이버에서 qtree/LUN 크기를 제한합니다. 새로운 것을 사용하세요 `limitVolumePoolSize` 해당 드라이버에서 Flexvol 크기를 제어하는 매개변수입니다. ("341호").
- 더 이상 사용되지 않는 igroup이 사용 중인 경우 정확한 LUN ID로 SCSI 스캔을 시작하기 위한 iSCSI 자체 복구 기능 추가("883호").
- 백엔드가 일시 중단 모드에 있을 때에도 볼륨 복제 및 크기 조정 작업이 허용되도록 지원이 추가되었습니다.
- Trident 컨트롤러에 대한 사용자 구성 로그 설정을 Trident 노드 포드로 전파할 수 있는 기능이 추가되었습니다.
- ONTAP 버전 9.15.1 이상에서 ONTAPI(ZAPI) 대신 기본적으로 REST를 사용하도록 Trident 에 지원이 추가되었습니다.
- 새로운 영구 볼륨에 대한 ONTAP 스토리지 백엔드에서 사용자 정의 볼륨 이름과 메타데이터에 대한 지원이 추가되었습니다.
- 강화되었습니다 `azure-netapp-files` NFS 마운트 옵션이 NFS 버전 4.x를 사용하도록 설정된 경우 기본적으로 스냅샷 디렉토리를 자동으로 활성화하는 (ANF) 드라이버입니다.
- NFS 볼륨에 대한 Bottlerocket 지원이 추가되었습니다.
- Google Cloud NetApp Volumes 에 대한 기술 미리 보기 지원이 추가되었습니다.

쿠버네티스

- Kubernetes 1.30에 대한 지원이 추가되었습니다.
- Trident DaemonSet이 시작 시 좀비 마운트와 잔여 추적 파일을 정리할 수 있는 기능이 추가되었습니다."883호").
- PVC 주석 추가 `trident.netapp.io/luksEncryption` LUKS 볼륨을 동적으로 가져오기 위해("849호").
- ANF 드라이버에 토폴로지 인식 기능이 추가되었습니다.
- Windows Server 2022 노드에 대한 지원이 추가되었습니다.

수정 사항

- 오래된 거래로 인해 Trident 설치가 실패하는 문제를 해결했습니다.
- Kubernetes에서 경고 메시지를 무시하도록 tridentctl을 수정했습니다."892호").
- Trident 컨트롤러 변경 `SecurityContextConstraint` 우선 순위 `0` ("887호").
- ONTAP 드라이버는 이제 20MiB 이하의 볼륨 크기를 허용합니다."문제[#885").
- ONTAP -SAN 드라이버의 크기 조정 작업 중 FlexVol 볼륨이 축소되는 것을 방지하기 위해 Trident 수정했습니다.
- NFS v4.1에서 ANF 볼륨 가져오기 실패 문제가 해결되었습니다.

24.02의 변경 사항

개선 사항

- Cloud Identity에 대한 지원이 추가되었습니다.
  - ANF를 사용한 AKS - Azure Workload Identity가 클라우드 ID로 사용됩니다.
  - FSxN이 있는 EKS - AWS IAM 역할이 클라우드 ID로 사용됩니다.

- EKS 콘솔에서 EKS 클러스터에 Trident 애드온으로 설치할 수 있는 지원이 추가되었습니다.
- iSCSI 자체 복구를 구성하고 비활성화하는 기능 추가("864호").
- AWS IAM 및 SecretsManager와의 통합을 활성화하고 Trident 백업을 사용하여 FSx 볼륨을 삭제할 수 있도록 ONTAP 드라이버에 Amazon FSx 개성을 추가했습니다."453호").

쿠버네티스

- Kubernetes 1.29에 대한 지원이 추가되었습니다.

수정 사항

- ACP가 활성화되지 않은 경우 고정 ACP 경고 메시지("866호").
- 복제본이 스냅샷과 연결되어 있는 경우 ONTAP 드라이버의 스냅샷 삭제 중에 복제본 분할을 수행하기 전에 10초 지연이 추가되었습니다.

사용 중단

- 다중 플랫폼 이미지 매니페스트에서 전체 증명 프레임워크를 제거했습니다.

23.10의 변경 사항

수정 사항

- ontap-nas 및 ontap-nas-flexgroup 스토리지 드라이버의 경우 새로 요청된 크기가 총 볼륨 크기보다 작을 경우 고정 볼륨 확장("834호").
- ontap-nas 및 ontap-nas-flexgroup 스토리지 드라이버에 대한 가져오기 중에 사용 가능한 볼륨 크기만 표시하도록 고정 볼륨 크기("722호").
- ONTAP -NAS-Economy에 대한 FlexVol 이름 변환이 수정되었습니다.
- 노드를 재부팅할 때 Windows 노드에서 Trident 초기화 문제가 해결되었습니다.

개선 사항

쿠버네티스

Kubernetes 1.28에 대한 지원이 추가되었습니다.

Trident

- azure-netapp-files 스토리지 드라이버와 함께 Azure Managed Identities(AMI)를 사용할 수 있는 지원이 추가되었습니다.
- ONTAP-SAN 드라이버에 대한 NVMe over TCP 지원이 추가되었습니다.
- 사용자가 백엔드를 일시 중지 상태로 설정한 경우 볼륨 프로비저닝을 일시 중지하는 기능이 추가되었습니다."558호").

23.07.1의 변경 사항

Kubernetes: 다운타임 없는 업그레이드를 지원하기 위해 데몬셋 삭제가 수정되었습니다."740호").

23.07의 변경 사항

수정 사항

쿠버네티스

- 종료 상태에 갇힌 오래된 포드를 무시하도록 고정된 Trident 업그레이드("740호").
- "transient-trident-version-pod" 정의에 허용 기능이 추가되었습니다."795호").

Trident

- 노드 스테이징 작업 중에 고스트 iSCSI 장치를 식별하고 수정하기 위해 LUN 속성을 가져올 때 LUN 일련 번호가 쿼리되도록 ONTAPI(ZAPI) 요청을 수정했습니다.
- 저장 드라이버 코드의 오류 처리가 고정되었습니다("816호").
- use-rest=true로 ONTAP 드라이버를 사용할 때 할당량 크기가 고정되었습니다.
- ontap-san-economy에서 LUN 복제본 생성이 고정되었습니다.
- 게시 정보 필드를 다음에서 되돌리기 `rawDevicePath` 에게 `devicePath` ; 채우기 및 복구(일부 경우)를 위한 논리가 추가되었습니다. `devicePath` 필드.

개선 사항

쿠버네티스

- 사전 프로비저닝된 스냅샷을 가져오는 기능이 추가되었습니다.
- 최소화된 배포 및 daemonset linux 권한("817호").

Trident

- 더 이상 "온라인" 볼륨과 스냅샷에 대한 상태 필드를 보고하지 않습니다.
- ONTAP 백엔드가 오프라인인 경우 백엔드 상태를 업데이트합니다."문제 #801" ,"#543").
- LUN 일련 번호는 항상 ControllerVolumePublish 워크플로우 동안 검색되고 게시됩니다.
- iSCSI 멀티패스 장치의 일련 번호와 크기를 확인하기 위한 추가 논리를 추가했습니다.
- iSCSI 볼륨에 대한 추가 검증을 통해 올바른 다중 경로 장치가 단계적으로 해제되었는지 확인합니다.

실험적 향상

ONTAP-SAN 드라이버에 대한 NVMe over TCP에 대한 기술 미리보기 지원이 추가되었습니다.

설명서

많은 조직 및 형식적 개선이 이루어졌습니다.

사용 중단

쿠버네티스

- v1beta1 스냅샷에 대한 지원이 제거되었습니다.

- CSI 이전 볼륨 및 스토리지 클래스에 대한 지원이 제거되었습니다.
- 지원되는 최소 Kubernetes 버전이 1.22로 업데이트되었습니다.

23.04의 변경 사항

> ONTAP-SAN-* 볼륨에 대한 강제 볼륨 분리는 Non-Graceful Node Shutdown 기능 게이트가 활성화된 Kubernetes 버전에서만 지원됩니다. 강제 분리는 설치 시 다음을 사용하여 활성화해야 합니다. `--enable-force-detach` Trident 설치 프로그램 플래그.

수정 사항

- 사양에 지정된 경우 설치를 위해 IPv6 로컬호스트를 사용하도록 Trident Operator를 수정했습니다.
- Trident Operator 클러스터 역할 권한이 번들 권한과 동기화되도록 수정되었습니다."799호").
- RWX 모드에서 여러 노드에 원시 블록 볼륨을 첨부하는 데 발생하는 문제가 해결되었습니다.
- SMB 볼륨에 대한 FlexGroup 복제 지원 및 볼륨 가져오기가 수정되었습니다.
- Trident 컨트롤러가 즉시 종료되지 않는 문제가 해결되었습니다."811호").
- ontap-san-* 드라이버로 프로비저닝된 지정된 LUN과 연관된 모든 igroup 이름을 나열하는 수정 사항이 추가되었습니다.
- 외부 프로세스가 완료될 때까지 실행할 수 있도록 수정 사항을 추가했습니다.
- s390 아키텍처에 대한 컴파일 오류가 수정되었습니다."537호").
- 볼륨 마운트 작업 중 잘못된 로깅 수준이 수정되었습니다."781호").
- 고정된 잠재적 유형 어설션 오류("802호").

개선 사항

- 쿠버네티스:
  - Kubernetes 1.27에 대한 지원이 추가되었습니다.
  - LUKS 볼륨 가져오기에 대한 지원이 추가되었습니다.
  - ReadWriteOncePod PVC 액세스 모드에 대한 지원이 추가되었습니다.
  - 비정상적인 노드 종료 시나리오 동안 ONTAP-SAN-* 볼륨에 대한 강제 분리 지원이 추가되었습니다.
  - 모든 ONTAP-SAN-* 볼륨은 이제 노드별 igroup을 사용합니다. LUN은 보안 태세를 개선하기 위해 해당 노드에 적극적으로 게시되는 동안에만 igroup에 매핑됩니다. Trident 활성 작업 부하에 영향을 미치지 않고 안전하다고 판단할 때 기존 볼륨은 기회적으로 새로운 igroup 체계로 전환됩니다."758호").
  - ONTAP -SAN-* 백엔드에서 사용되지 않는 Trident 관리 igroup을 정리하여 Trident 보안을 개선했습니다.
- Amazon FSx 사용하여 ontap-nas-economy 및 ontap-nas-flexgroup 스토리지 드라이버에 SMB 볼륨에 대한 지원이 추가되었습니다.
- ontap-nas, ontap-nas-economy 및 ontap-nas-flexgroup 스토리지 드라이버를 사용하여 SMB 공유에 대한 지원이 추가되었습니다.
- arm64 노드에 대한 지원이 추가되었습니다("732호").
- API 서버를 먼저 비활성화하여 Trident 종료 절차를 개선했습니다."811호").

- Makefile에 Windows 및 arm64 호스트에 대한 크로스 플랫폼 빌드 지원이 추가되었습니다. BUILD.md를 참조하세요.

사용 중단

Kubernetes: ontap-san 및 ontap-san-economy 드라이버를 구성할 때 백엔드 범위의 igroup이 더 이상 생성되지 않습니다."758호").

23.01.1의 변경 사항

수정 사항

- 사양에 지정된 경우 설치를 위해 IPv6 로컬호스트를 사용하도록 Trident Operator를 수정했습니다.
- Trident Operator 클러스터 역할 권한이 번들 권한과 동기화되도록 수정되었습니다."799호" .
- 외부 프로세스가 완료될 때까지 실행할 수 있도록 수정 사항을 추가했습니다.
- RWX 모드에서 여러 노드에 원시 블록 볼륨을 첨부하는 데 발생하는 문제가 해결되었습니다.
- SMB 볼륨에 대한 FlexGroup 복제 지원 및 볼륨 가져오기가 수정되었습니다.

23.01의 변경 사항

> Kubernetes 1.27가 이제 Trident 에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident 업그레이드하세요.

수정 사항

- Kubernetes: Helm을 통해 Trident 설치를 수정하기 위해 Pod 보안 정책 생성을 제외하는 옵션이 추가되었습니다 ."783호, 794호").

개선 사항

쿠버네티스

- Kubernetes 1.26에 대한 지원이 추가되었습니다.
- 전반적인 Trident RBAC 리소스 활용도 향상("757호").
- 호스트 노드에서 끊어지거나 오래된 iSCSI 세션을 감지하고 수정하는 자동화 기능이 추가되었습니다.
- LUKS 암호화 볼륨 확장에 대한 지원이 추가되었습니다.
- Kubernetes: LUKS 암호화 볼륨에 대한 자격 증명 회전 지원이 추가되었습니다.

Trident

- ontap-nas 스토리지 드라이버에 Amazon FSx for NetApp ONTAP 사용한 SMB 볼륨 지원이 추가되었습니다.
- SMB 볼륨을 사용할 때 NTFS 권한에 대한 지원이 추가되었습니다.
- CVS 서비스 수준을 갖춘 GCP 볼륨의 스토리지 풀에 대한 지원이 추가되었습니다.
- ontap-nas-flexgroup 스토리지 드라이버를 사용하여 FlexGroup을 생성할 때 flexgroupAggregateList를 선택적으로 사용할 수 있는 지원이 추가되었습니다.
- 여러 FlexVol 볼륨을 관리할 때 ontap-nas-economy 스토리지 드라이버의 성능이 향상되었습니다.

- 모든 ONTAP NAS 스토리지 드라이버에 대해 dataLIF 업데이트가 활성화되었습니다.
- 호스트 노드 OS를 반영하도록 Trident Deployment 및 DaemonSet 명명 규칙을 업데이트했습니다.

사용 중단

- Kubernetes: 지원되는 최소 Kubernetes 버전이 1.21로 업데이트되었습니다.
- DataLIF는 더 이상 구성할 때 지정되지 않아야 합니다. `ontap-san` 또는 `ontap-san-economy` 운전자.

22.10의 변경 사항

- Trident 22.10으로 업그레이드하기 전에 다음의 중요 정보를 꼭 읽어보세요.*

> ** Trident 22.10**에 대한 중요 정보****
>
> - Kubernetes 1.25가 이제 Trident 에서 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Trident 22.10으로 업그레이드해야 합니다.
> - Trident 이제 SAN 환경에서 다중 경로 구성 사용을 엄격하게 적용하며 권장 값은 다음과 같습니다. `find_multipaths: no` multipath.conf 파일에서.
>
> 비다중경로 구성 사용 또는 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트가 실패합니다. Trident 다음을 사용할 것을 권장했습니다. `find_multipaths: no` 21.07 릴리스 이후.

수정 사항

- ONTAP 백엔드를 사용하여 생성된 특정 문제가 수정되었습니다. `credentials` 22.07.0 업그레이드 중 필드가 온라인 상태가 되지 않음("759호").
- **Docker:** 일부 환경에서 Docker 볼륨 플러그인이 시작되지 않는 문제를 해결했습니다."548호" 그리고"760호").
- 보고 노드에 속한 dataLIF의 하위 집합만 게시되도록 하기 위해 ONTAP SAN 백엔드에 특정한 SLM 문제가 수정되었습니다.
- 볼륨을 연결할 때 불필요한 iSCSI LUN 검색이 발생하는 성능 문제가 해결되었습니다.
- Trident iSCSI 워크플로 내에서 세분화된 재시도를 제거하여 빠르게 실패하고 외부 재시도 간격을 줄였습니다.
- 해당 멀티패스 장치가 이미 플러시된 경우 iSCSI 장치를 플러시할 때 오류가 반환되는 문제가 해결되었습니다.

개선 사항

- 쿠버네티스:
  - Kubernetes 1.25에 대한 지원이 추가되었습니다. Kubernetes 1.25로 업그레이드하기 전에 Trident 22.10으로 업그레이드해야 합니다.
  - Trident 배포 및 DaemonSet에 별도의 ServiceAccount, ClusterRole 및 ClusterRoleBinding을 추가하여 향후 권한 향상이 가능해졌습니다.
  - 추가 지원"크로스 네임스페이스 볼륨 공유" .
- 올 Trident `ontap-*` 스토리지 드라이버는 이제 ONTAP REST API와 함께 작동합니다.
- 새로운 연산자 yaml이 추가되었습니다.(`bundle_post_1_25.yaml`) 없이 `PodSecurityPolicy`

Kubernetes 1.25를 지원합니다.

- 추가됨"LUKS 암호화 볼륨 지원" ~을 위한 `ontap-san` 그리고 `ontap-san-economy` 스토리지 드라이버.
- Windows Server 2019 노드에 대한 지원이 추가되었습니다.
- 추가됨"Windows 노드에서 SMB 볼륨 지원" 통해 `azure-netapp-files` 저장 드라이버.
- ONTAP 드라이버에 대한 자동 MetroCluster 전환 감지 기능이 이제 일반적으로 사용 가능합니다.

사용 중단

- **Kubernetes:** 지원되는 최소 Kubernetes 버전이 1.20으로 업데이트되었습니다.
- Astra Data Store(ADS) 드라이버를 제거했습니다.
- 지원이 제거되었습니다. `yes` 그리고 `smart` 옵션 `find_multipaths` iSCSI에 대한 워커 노드 다중 경로를 구성할 때.

22.07의 변경 사항

수정 사항

쿠버네티스

- Helm이나 Trident Operator를 사용하여 Trident 구성할 때 노드 선택기의 부울 및 숫자 값을 처리하는 문제가 해결되었습니다. ("GitHub 이슈 #700")
- CHAP가 아닌 경로에서 발생하는 오류를 처리하는 데 발생하는 문제를 해결하여 kubelet이 실패할 경우 다시 시도하도록 했습니다. "GitHub 이슈 #736")

개선 사항

- CSI 이미지의 기본 레지스트리로 k8s.gcr.io에서 registry.k8s.io로 전환
- ONTAP-SAN 볼륨은 이제 노드별 igroup을 사용하고 해당 노드에 적극적으로 게시되는 동안에만 LUN을 igroup에 매핑하여 보안 태세를 강화합니다. Trident 활성 작업 부하에 영향을 주지 않고 안전하다고 판단하면 기존 볼륨은 새로운 igroup 체계로 기회적으로 전환됩니다.
- PriorityClass 소비가 기본적으로 제한되는 경우 Trident DaemonSet이 예약되도록 하기 위해 Trident 설치에 ResourceQuota를 포함했습니다.
- Azure NetApp Files 드라이버에 네트워크 기능에 대한 지원이 추가되었습니다. ("GitHub 이슈 #717")
- ONTAP 드라이버에 MetroCluster 전환 감지 기능을 자동으로 제공하는 기술 미리보기 기능이 추가되었습니다. ("GitHub 이슈 #228")

사용 중단

- **Kubernetes:** 지원되는 최소 Kubernetes 버전이 1.19로 업데이트되었습니다.
- 백엔드 구성에서는 더 이상 단일 구성에서 여러 인증 유형을 허용하지 않습니다.

이사

- AWS CVS 드라이버(22.04부터 더 이상 사용되지 않음)가 제거되었습니다.
- 쿠버네티스

- 노드 포드에서 불필요한 SYS_ADMIN 기능을 제거했습니다.
- NFS/iSCSI 서비스가 워커 노드에서 사용 가능한지 최선을 다해 확인하기 위해 nodeprep을 간단한 호스트 정보와 활성 서비스 검색으로 줄입니다.

설명서

새로운"포드 보안 표준" (PSS) 섹션에 Trident 설치 시 활성화된 권한에 대한 자세한 내용이 추가되었습니다.

22.04의 변경 사항

NetApp 제품과 서비스를 지속적으로 개선하고 향상시키고 있습니다. Trident 의 최신 기능은 다음과 같습니다. 이전 릴리스의 경우 다음을 참조하세요. "이전 버전의 문서" .

> 이전 Trident 릴리스에서 업그레이드하고 Azure NetApp Files 사용하는 경우`location` config 매개변수는 이제 필수 싱글톤 필드입니다.

수정 사항

- iSCSI 이니시에이터 이름 구문 분석이 개선되었습니다. ("GitHub 이슈 #681")
- CSI 스토리지 클래스 매개변수가 허용되지 않는 문제가 해결되었습니다. ("GitHub 이슈 #598")
- Trident CRD에서 중복 키 선언을 수정했습니다. ("GitHub 이슈 #671")
- 부정확한 CSI 스냅샷 로그를 수정했습니다. ("GitHub 이슈 #629"))
- 삭제된 노드에서 볼륨을 게시 취소하는 문제가 해결되었습니다. ("GitHub 이슈 #691")
- 블록 장치에서 파일 시스템 불일치 처리 기능이 추가되었습니다. ("GitHub 이슈 #656")
- 자동 지원 이미지를 설정할 때 발생하는 문제가 해결되었습니다. `imageRegistry` 설치 중에 플래그를 지정합니다. ("GitHub 이슈 #715")
- Azure NetApp Files 드라이버가 여러 내보내기 규칙이 있는 볼륨을 복제하지 못하는 문제가 해결되었습니다.

개선 사항

- Trident의 보안 엔드포인트에 대한 인바운드 연결에는 이제 최소 TLS 1.3이 필요합니다. ("GitHub 이슈 #698")
- Trident 이제 보안 엔드포인트의 응답에 HSTS 헤더를 추가합니다.
- Trident 이제 Azure NetApp Files Unix 권한 기능을 자동으로 활성화하려고 시도합니다.
- **Kubernetes**: Trident 데몬셋이 이제 시스템 노드 중요 우선 순위 클래스에서 실행됩니다. ("GitHub 이슈 #694")

이사

E-시리즈 드라이버(20.07부터 비활성화됨)가 제거되었습니다.

22.01.1의 변경 사항

수정 사항

- 삭제된 노드에서 볼륨을 게시 취소하는 문제가 해결되었습니다. ("GitHub 이슈 #691")

- ONTAP API 응답에서 집계 공간에 대한 nil 필드에 액세스할 때 발생하는 패닉을 수정했습니다.

22.01.0의 변경 사항

수정 사항

- **Kubernetes:** 대규모 클러스터의 노드 등록 백오프 재시도 시간을 늘립니다.
- azure-netapp-files 드라이버가 동일한 이름을 가진 여러 리소스로 인해 혼동될 수 있는 문제가 해결되었습니다.
- ONTAP SAN IPv6 DataLIF는 이제 대괄호로 지정된 경우에도 작동합니다.
- 이미 가져온 볼륨을 가져오려고 하면 EOF가 반환되고 PVC가 보류 상태로 남는 문제가 해결되었습니다. ("GitHub 이슈 #489")
- SolidFire 볼륨에서 스냅샷이 32개 이상 생성되면 Trident 성능이 저하되는 문제가 해결되었습니다.
- SSL 인증서 생성에서 SHA-1을 SHA-256으로 대체했습니다.
- 중복된 리소스 이름을 허용하고 작업을 단일 위치로 제한하도록 Azure NetApp Files 드라이버를 수정했습니다.
- 중복된 리소스 이름을 허용하고 작업을 단일 위치로 제한하도록 Azure NetApp Files 드라이버를 수정했습니다.

개선 사항

- Kubernetes 개선 사항:
  - Kubernetes 1.23에 대한 지원이 추가되었습니다.
  - Trident Operator 또는 Helm을 통해 Trident Pod를 설치하면 일정 옵션을 추가할 수 있습니다. ("GitHub 이슈 #651")
- GCP 드라이버에서 지역 간 볼륨을 허용합니다. ("GitHub 이슈 #633")
- Azure NetApp Files 볼륨에 'unixPermissions' 옵션에 대한 지원이 추가되었습니다. ("GitHub 이슈 #666")

사용 중단

Trident REST 인터페이스는 127.0.0.1 또는 [::1] 주소에서만 수신하고 서비스할 수 있습니다.

21.10.1의 변경 사항

> v21.10.0 릴리스에는 노드를 제거한 다음 Kubernetes 클러스터에 다시 추가하면 Trident 컨트롤러가 CrashLoopBackOff 상태가 될 수 있는 문제가 있습니다. 이 문제는 v21.10.1(GitHub 문제 669)에서 해결되었습니다.

수정 사항

- GCP CVS 백엔드에서 볼륨을 가져올 때 발생할 수 있는 경쟁 조건으로 인해 가져오기에 실패하는 문제를 해결했습니다.
- 노드를 제거한 후 Kubernetes 클러스터에 다시 추가하면 Trident 컨트롤러가 CrashLoopBackOff 상태가 되는 문제를 해결했습니다(GitHub 문제 669).
- SVM 이름이 지정되지 않은 경우 SVM을 더 이상 검색하지 못하는 문제가 해결되었습니다(GitHub 문제 612).

21.10.0의 변경 사항

수정 사항

- XFS 볼륨의 복제본을 소스 볼륨과 동일한 노드에 마운트할 수 없는 문제가 해결되었습니다(GitHub 문제 514).
- Trident 종료 시 치명적인 오류를 기록하는 문제가 해결되었습니다(GitHub 문제 597).
- Kubernetes 관련 수정 사항:
  - 스냅샷을 생성할 때 볼륨의 사용된 공간을 최소 restoreSize로 반환합니다. `ontap-nas` 그리고 `ontap-nas-flexgroup` 드라이버(GitHub 이슈 645).
  - 문제가 해결되었습니다. `Failed to expand filesystem` 볼륨 크기 조정 후 오류가 기록되었습니다(GitHub 문제 560).
  - 포드가 끼일 수 있는 문제가 해결되었습니다. `Terminating` 상태(GitHub 이슈 572).
  - 해결된 사례 `ontap-san-economy` FlexVol 스냅샷 LUN으로 가득 차 있을 수 있습니다(GitHub 문제 533).
  - 다른 이미지로 인한 사용자 지정 YAML 설치 프로그램 문제가 해결되었습니다(GitHub 문제 613).
  - 스냅샷 크기 계산이 고정되었습니다(GitHub 문제 611).
  - 모든 Trident 설치 프로그램이 일반 Kubernetes를 OpenShift로 식별하는 문제가 해결되었습니다(GitHub 문제 639).
  - Kubernetes API 서버에 접속할 수 없는 경우 조정을 중지하도록 Trident 연산자를 수정했습니다(GitHub 문제 599).

개선 사항

- 추가 지원 `unixPermissions` GCP-CVS 성능 볼륨에 대한 옵션입니다.
- GCP에서 600GiB~1TiB 범위의 규모 최적화된 CVS 볼륨에 대한 지원이 추가되었습니다.
- Kubernetes 관련 개선 사항:
  - Kubernetes 1.22에 대한 지원이 추가되었습니다.
  - Kubernetes 1.22에서 Trident 연산자와 Helm 차트가 작동하도록 했습니다(GitHub 문제 628).
  - 운영자 이미지 추가 `tridentctl` images 명령(GitHub 이슈 570).

실험적 개선 사항

- 볼륨 복제에 대한 지원이 추가되었습니다. `ontap-san` 운전사.
- REST 지원에 대한 *기술 미리보기*가 추가되었습니다. `ontap-nas-flexgroup` , `ontap-san` , 그리고 `ontap-nas-economy` 운전자.

알려진 문제

알려진 문제는 제품을 성공적으로 사용하는 데 방해가 될 수 있는 문제를 나타냅니다.

- Trident 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드하는 경우 values.yaml을 업데이트하여 설정해야 합니다. `excludePodSecurityPolicy` 에게 `true` 또는 추가 `--set excludePodSecurityPolicy=true` 에게 `helm upgrade` 클러스터를 업그레이드하기 전에 명령을 실행하세요.

- Trident 이제 공백을 적용합니다. `fsType` (`fsType=""`) 볼륨에 없는 경우 `fsType` StorageClass에 지정되어 있습니다. Kubernetes 1.17 이상을 사용하는 경우 Trident 빈칸을 제공하는 것을 지원합니다. `fsType` NFS 볼륨의 경우. iSCSI 볼륨의 경우 다음을 설정해야 합니다. `fsType` StorageClass에서 적용할 때 `fsGroup` 보안 컨텍스트를 사용합니다.
- 여러 Trident 인스턴스에서 백엔드를 사용하는 경우 각 백엔드 구성 파일에는 다른 내용이 있어야 합니다. `storagePrefix` ONTAP 백엔드에 대한 값을 사용하거나 다른 값을 사용하세요. `TenantName` SolidFire 백엔드용. Trident 다른 Trident 인스턴스가 생성한 볼륨을 감지할 수 없습니다. ONTAP 또는 SolidFire 백엔드에서 기존 볼륨을 생성하려는 시도는 성공합니다. Trident 볼륨 생성을 멱등 작업으로 처리하기 때문입니다. 만약에 `storagePrefix` 또는 `TenantName` 다르지 않습니다. 동일한 백엔드에서 생성된 볼륨의 경우 이름 충돌이 발생할 수 있습니다.
- Trident 설치할 때(사용 `tridentctl` 또는 Trident Operator)를 사용하여 `tridentctl` Trident 관리하려면 다음을 확인해야 합니다. `KUBECONFIG` 환경 변수가 설정되었습니다. 이는 Kubernetes 클러스터를 표시하는 데 필요합니다. `tridentctl` 반대해야 합니다. 여러 Kubernetes 환경에서 작업할 때는 다음을 확인해야 합니다. `KUBECONFIG` 파일의 출처가 정확합니다.
- iSCSI PV에 대한 온라인 공간 회수를 수행하려면 작업자 노드의 기본 OS에서 볼륨에 마운트 옵션을 전달해야 할 수 있습니다. 이는 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 인스턴스에 해당하며 여기에는 다음이 필요합니다. `discard` "마운트 옵션" ; discard mountOption이 포함되어 있는지 확인하세요.[`StorageClass` ^] 온라인 블록 삭제를 지원합니다.
- Kubernetes 클러스터당 Trident 인스턴스가 두 개 이상 있는 경우 Trident 다른 인스턴스와 통신할 수 없고 해당 인스턴스가 생성한 다른 볼륨을 검색할 수 없습니다. 이로 인해 클러스터 내에서 두 개 이상의 인스턴스가 실행되는 경우 예상치 못한 잘못된 동작이 발생합니다. Kubernetes 클러스터당 Trident 인스턴스는 하나만 있어야 합니다.
- 트라이던트 기반이라면 `StorageClass` Trident 오프라인 상태일 때 Kubernetes에서 객체가 삭제되고, Trident 가 다시 온라인 상태가 되면 데이터베이스에서 해당 스토리지 클래스를 제거하지 않습니다. 다음 저장소 클래스를 사용하여 삭제해야 합니다. `tridentctl` 또는 REST API.
- 사용자가 해당 PVC를 삭제하기 전에 Trident 에서 프로비저닝한 PV를 삭제하는 경우, Trident 자동으로 백업 볼륨을 삭제하지 않습니다. 볼륨을 제거해야 합니다. `tridentctl` 또는 REST API.
- ONTAP 각 프로비저닝 요청에 대한 집계 세트가 고유하지 않은 한 한 번에 두 개 이상의 FlexGroup 동시에 프로비저닝할 수 없습니다.
- IPv6를 통해 Trident 사용하는 경우 다음을 지정해야 합니다. `managementLIF` 그리고 `dataLIF` 백엔드 정의에서 대괄호로 묶어서 표현합니다. 예를 들어, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]` .

  > ⓘ 지정할 수 없습니다 `dataLIF` ONTAP SAN 백엔드에서. Trident 사용 가능한 모든 iSCSI LIF를 검색하고 이를 사용하여 다중 경로 세션을 설정합니다.

- 사용하는 경우 `solidfire-san` OpenShift 4.5 드라이버를 사용하는 경우 기본 작업자 노드가 MD5를 CHAP 인증 알고리즘으로 사용하는지 확인하세요. Element 12.7에서는 보안 FIPS 호환 CHAP 알고리즘 SHA1, SHA-256, SHA3-256을 사용할 수 있습니다.

더 많은 정보를 찾아보세요

- "Trident GitHub"
- "Trident 블로그"

이전 버전의 문서

Trident 25.06을 실행하지 않는 경우 이전 릴리스에 대한 설명서를 사용할 수 있습니다."Trident

지원 수명주기" .

- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"
- "Trident 22.10"

알려진 문제

알려진 문제는 이 제품 릴리스를 성공적으로 사용하는 데 방해가 될 수 있는 문제를 나타냅니다.

현재 릴리스에는 다음과 같은 알려진 문제가 있습니다.

대용량 파일의 **Restic** 백업을 복원하는 데 실패할 수 있습니다.

Restic을 사용하여 만든 Amazon S3 백업에서 30GB 이상의 파일을 복원하는 경우 복원 작업이 실패할 수 있습니다. 해결 방법으로, Kopia를 데이터 이동 도구로 사용하여 데이터를 백업합니다(Kopia는 백업을 위한 기본 데이터 이동 도구입니다). 참조하다 "Trident Protect를 사용하여 애플리케이션 보호" 지침을 보려면.

시작하기

Trident 에 대해 알아보세요

Trident 에 대해 알아보세요

Trident NetApp 에서 유지 관리하는 완벽하게 지원되는 오픈 소스 프로젝트입니다. 이 기능은 CSI(Container Storage Interface)와 같은 업계 표준 인터페이스를 사용하여 컨테이너화된 애플리케이션의 지속성 요구 사항을 충족하는 데 도움이 되도록 설계되었습니다.

Trident 란 무엇인가요?

Netapp Trident FAS 하면 온프레미스 ONTAP 클러스터(AFF , NetApp , ASA), ONTAP Select, Cloud Volumes ONTAP, Element 소프트웨어(NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP, Google Cloud의 Cloud Volumes Service 등 퍼블릭 클라우드나 온프레미스에서 모든 인기 있는 NetApp 스토리지 플랫폼의 스토리지 리소스를 사용하고 관리할 수 있습니다.

Trident 는 기본적으로 통합되는 컨테이너 스토리지 인터페이스(CSI) 호환 동적 스토리지 오케스트레이터입니다. "쿠버네티스" . Trident 클러스터의 각 워커 노드에 있는 노드 포드와 단일 컨트롤러 포드로 실행됩니다. 참조하다 "Trident 아키텍처" 자세한 내용은.

Trident 또한 NetApp 스토리지 플랫폼을 위한 Docker 생태계와의 직접적인 통합을 제공합니다. NetApp Docker Volume Plugin(nDVP)은 스토리지 플랫폼에서 Docker 호스트로의 스토리지 리소스 프로비저닝 및 관리를 지원합니다. 참조하다 "Docker에 Trident 배포" 자세한 내용은.

Kubernetes를 처음 사용하는 경우 다음 사항에 익숙해져야 합니다."쿠버네티스 개념 및 도구" .

NetApp 제품과 Kubernetes 통합

NetApp 스토리지 제품 포트폴리오는 Kubernetes 클러스터의 여러 측면과 통합되어 고급 데이터 관리 기능을 제공하며, 이를 통해 Kubernetes 배포의 기능, 역량, 성능 및 가용성이 향상됩니다.

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP"NetApp ONTAP 스토리지 운영 체제를 기반으로 파일 시스템을 실행하고 실행할 수 있는 완전 관리형 AWS 서비스입니다.

Azure NetApp Files

"Azure NetApp Files"NetApp 기반의 엔터프라이즈급 Azure 파일 공유 서비스입니다. NetApp 에서 기대하는 성능과 풍부한 데이터 관리 기능을 통해 Azure에서 가장 까다로운 파일 기반 워크로드를 기본적으로 실행할 수 있습니다.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP"클라우드에서 ONTAP 데이터 관리 소프트웨어를 실행하는 소프트웨어 전용 스토리지 어플라이언스입니다.

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes"Google Cloud의 완전 관리형 파일 저장 서비스로, 고성능의 엔터프라이즈급 파일 저장을 제공합니다.

엘리먼트 소프트웨어

"요소"스토리지 관리자가 성능을 보장하고 간소화되고 효율적인 스토리지 공간을 확보함으로써 작업 부하를 통합할 수 있도록 지원합니다.

NetApp HCI

"NetApp HCI"일상적인 작업을 자동화하고 인프라 관리자가 더 중요한 기능에 집중할 수 있도록 하여 데이터 센터의 관리와 규모를 간소화합니다.

Trident 기본 NetApp HCI 스토리지 플랫폼에 직접 컨테이너화된 애플리케이션의 스토리지 장치를 프로비저닝하고 관리할 수 있습니다.

NetApp ONTAP

"NetApp ONTAP"NetApp 다중 프로토콜 통합 스토리지 운영 체제로, 모든 애플리케이션에 고급 데이터 관리 기능을 제공합니다.

ONTAP 시스템은 올플래시, 하이브리드 또는 올HDD 구성을 갖추고 있으며 온프레미스 FAS, AFA 및 ASA 클러스터, ONTAP Select, Cloud Volumes ONTAP 다양한 배포 모델을 제공합니다. Trident 이러한 ONTAP 배포 모델을 지원합니다.

Trident 아키텍처

Trident 클러스터의 각 워커 노드에 있는 노드 포드와 단일 컨트롤러 포드로 실행됩니다. Trident 볼륨을 마운트하려는 모든 호스트에서 노드 포드를 실행해야 합니다.

컨트롤러 포드와 노드 포드 이해

Trident 단일로 배치됩니다.Trident 컨트롤러 포드 그리고 하나 이상Trident 노드 포드 Kubernetes 클러스터에서 표준 Kubernetes _CSI Sidecar Containers_를 사용하여 CSI 플러그인 배포를 간소화합니다. "Kubernetes CSI 사이드카 컨테이너" Kubernetes Storage 커뮤니티에서 유지 관리됩니다.

쿠버네티스"노드 선택기" 그리고"관용과 오염" 특정 노드나 선호하는 노드에서 포드가 실행되도록 제한하는 데 사용됩니다. Trident 설치 중에 컨트롤러와 노드 포드에 대한 노드 선택기와 허용 범위를 구성할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.

- 노드 플러그인은 저장소를 노드에 연결하는 작업을 처리합니다.

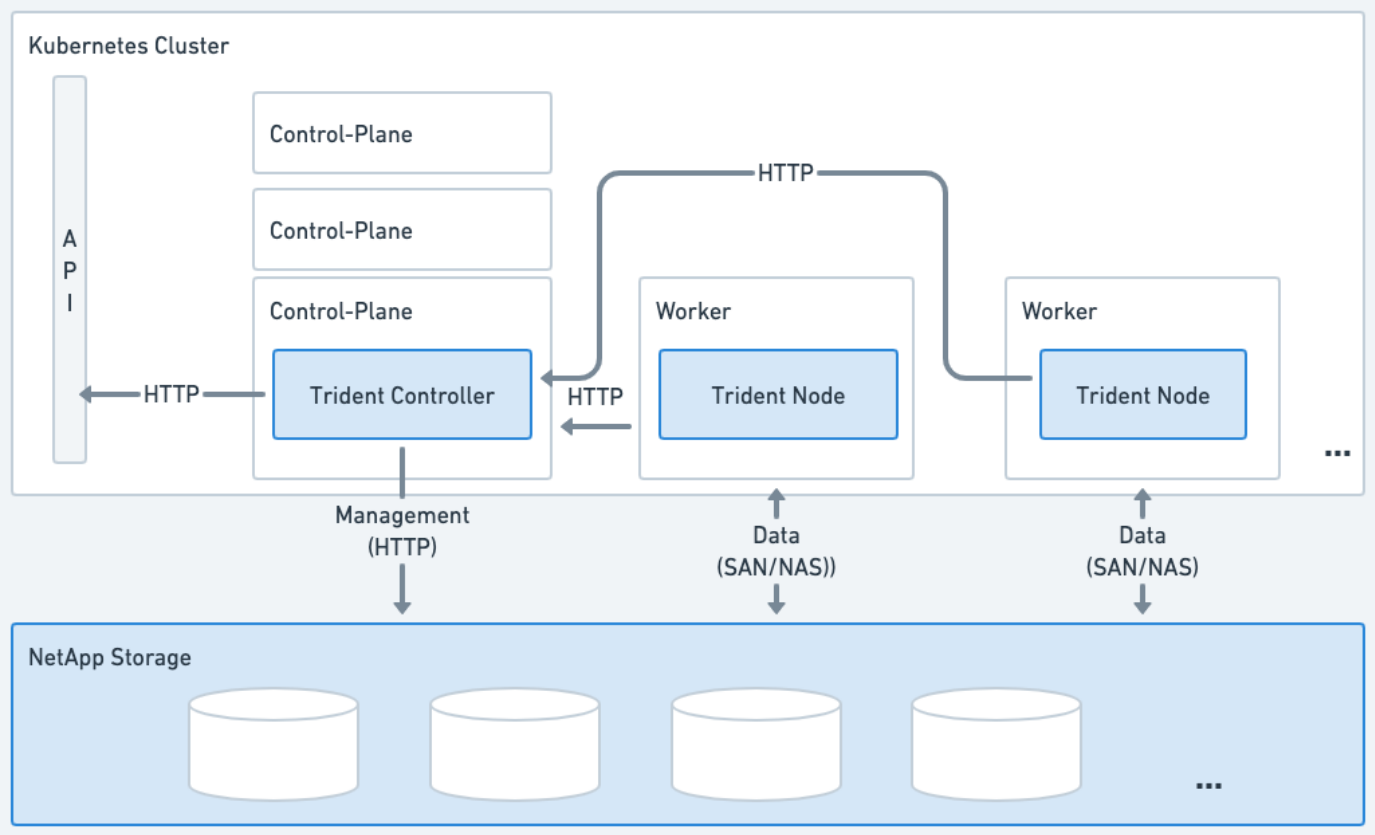


그림 **1. Kubernetes** 클러스터에 **Trident** 배포

Trident 컨트롤러 포드

Trident Controller Pod는 CSI Controller 플러그인을 실행하는 단일 Pod입니다.

- NetApp 스토리지의 볼륨 프로비저닝 및 관리를 담당합니다.
- Kubernetes 배포로 관리됨
- 설치 매개변수에 따라 제어 평면이나 작업자 노드에서 실행할 수 있습니다.

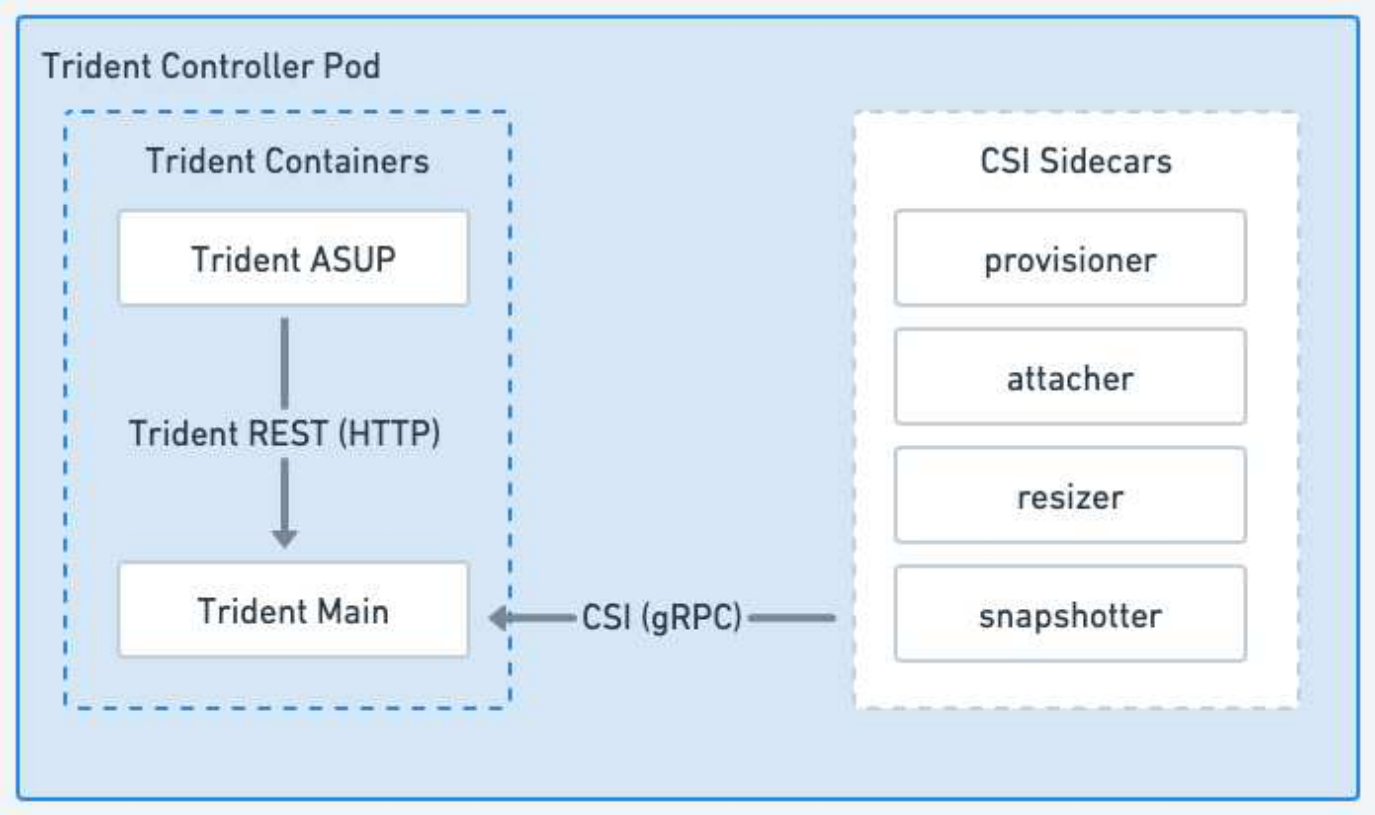


그림 **2. Trident** 컨트롤러 포드 다이어그램

Trident 노드 포드

Trident Node Pod는 CSI Node 플러그인을 실행하는 특권 Pod입니다.

- 호스트에서 실행되는 Pod에 대한 스토리지 마운트 및 마운트 해제를 담당합니다.
- Kubernetes DaemonSet에서 관리됨
- NetApp 스토리지를 마운트할 모든 노드에서 실행해야 함

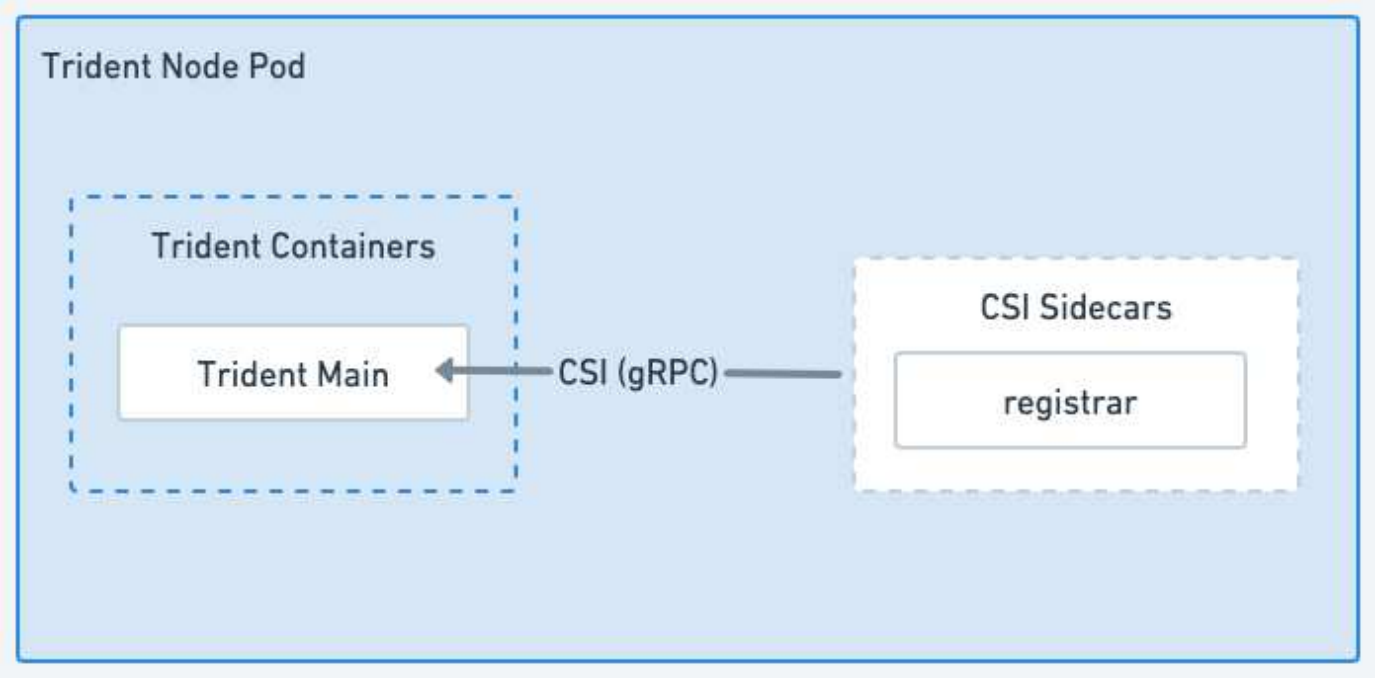


그림 **3. Trident** 노드 포드 다이어그램

지원되는 **Kubernetes** 클러스터 아키텍처

Trident 다음과 같은 Kubernetes 아키텍처에서 지원됩니다.

쿠버네티스 클러스터 아키텍처	지원됨	기본 설치
단일 마스터, 컴퓨팅	예	예
다중 마스터, 컴퓨팅	예	예
주인, `etcd` , 계산하다	예	예
마스터, 인프라, 컴퓨팅	예	예

개념

프로비저닝

Trident 의 프로비저닝은 두 가지 주요 단계로 구성됩니다. 첫 번째 단계에서는 스토리지 클래스를 적합한 백엔드 스토리지 풀 세트와 연결하며 프로비저닝 전에 필요한 준비 작업으로 진행됩니다. 두 번째 단계에는 볼륨 생성 자체가 포함되며 보류 중인 볼륨의 스토리지 클래스와 연관된 스토리지 풀 중에서 스토리지 풀을 선택해야 합니다.

스토리지 클래스 연결

백엔드 스토리지 풀을 스토리지 클래스와 연결하는 것은 스토리지 클래스의 요청된 속성과 해당 속성 모두에 의존합니다. `storagePools` , `additionalStoragePools` , 그리고 `excludeStoragePools` 기울기. 스토리지 클래스를 생성하면 Trident 각 백엔드에서 제공하는 속성과 풀을 스토리지 클래스에서 요청한 속성과 풀과 비교합니다.

스토리지 풀의 속성과 이름이 요청된 모든 속성 및 풀 이름과 일치하면 Trident 해당 스토리지 풀을 해당 스토리지 클래스에 적합한 스토리지 풀 세트에 추가합니다. 또한 Trident 다음에 나열된 모든 스토리지 풀을 추가합니다. `additionalStoragePools` 해당 속성이 저장 클래스의 요청된 속성 중 일부 또는 전부를 충족하지 않더라도 해당 집합에 대한 목록을 작성합니다. 당신은 사용해야 합니다 `excludeStoragePools` 스토리지 클래스에 대한 스토리지 풀 사용을 재정의하고 제거하기 위한 목록입니다. Trident 새로운 백엔드를 추가할 때마다 비슷한 프로세스를 수행하여 스토리지 풀이 기존 스토리지 클래스의 스토리지 풀을 충족하는지 확인하고 제외된 것으로 표시된 스토리지 풀을 제거합니다.

볼륨 생성

그런 다음 Trident 스토리지 클래스와 스토리지 풀 간의 연결을 사용하여 볼륨을 프로비저닝할 위치를 결정합니다. 볼륨을 생성하면 Trident 먼저 해당 볼륨의 스토리지 클래스에 대한 스토리지 풀 세트를 가져오고, 볼륨에 대한 프로토콜을 지정하면 Trident 요청된 프로토콜을 제공할 수 없는 스토리지 풀을 제거합니다(예: NetApp HCI/ SolidFire 백엔드는 파일 기반 볼륨을 제공할 수 없고 ONTAP NAS 백엔드는 블록 기반 볼륨을 제공할 수 없습니다). Trident 볼륨을 균등하게 분배하기 위해 결과 집합의 순서를 무작위로 지정한 다음 이를 반복하여 각 스토리지 풀에 볼륨을 차례로 프로비저닝하려고 시도합니다. 둘 중 하나에 성공하면 성공적으로 반환하고, 프로세스에서 발생한 모든 실패를 기록합니다. Trident 요청된 스토리지 클래스 및 프로토콜에 사용 가능한 모든 스토리지 풀을 프로비저닝하지 못한 경우에만 실패를 반환합니다.

볼륨 스냅샷

Trident 드라이버에 대한 볼륨 스냅샷을 생성하는 방법에 대해 자세히 알아보세요.

볼륨 스냅샷 생성에 대해 알아보세요

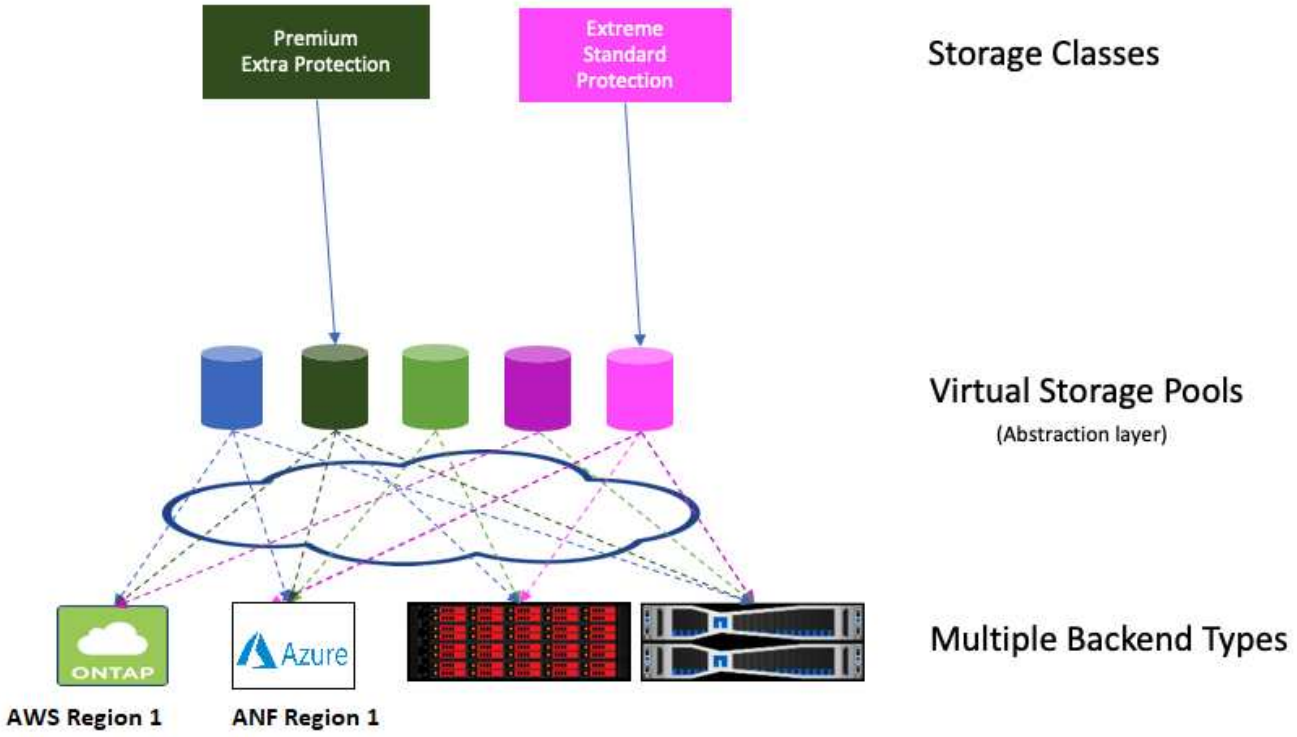
- 를 위해 `ontap-nas` , `ontap-san` , `gcp-cvs` , 그리고 `azure-netapp-files` 드라이버, 각 영구 볼륨(PV)은 FlexVol volume 에 매핑됩니다. 결과적으로 볼륨 스냅샷은 NetApp 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁사의 스냅샷 기술보다 더 뛰어난 안정성, 확장성, 복구 가능성 및 성능을 제공합니다. 이러한 스냅샷 복사본은 복사본을 만드는 데 필요한 시간과 저장 공간 측면에서 매우 효율적입니다.
- 를 위해 `ontap-nas-flexgroup` 드라이버, 각 영구 볼륨(PV)은 FlexGroup 에 매핑됩니다. 결과적으로 볼륨 스냅샷은 NetApp FlexGroup 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁사의 스냅샷 기술보다 더 뛰어난 안정성, 확장성, 복구 가능성 및 성능을 제공합니다. 이러한 스냅샷 복사본은 복사본을 만드는 데 필요한 시간과 저장 공간 측면에서 매우 효율적입니다.
- 를 위해 `ontap-san-economy` 드라이버, PV는 공유 FlexVol 볼륨에 생성된 LUN에 매핑됩니다. PV의 VolumeSnapshot은 연관된 LUN의 FlexClone을 수행하여 구현됩니다. ONTAP FlexClone 기술을 사용하면 아무리 큰 데이터 세트라도 거의 즉시 복사본을 생성할 수 있습니다. 복사본은 부모와 데이터 블록을 공유하므로 메타데이터에 필요한 것 외에는 저장 공간을 사용하지 않습니다.
- 를 위해 `solidfire-san` 드라이버, 각 PV는 NetApp Element 소프트웨어/ NetApp HCI 클러스터에서 생성된 LUN에 매핑됩니다. VolumeSnapshots는 기본 LUN의 요소 스냅샷으로 표현됩니다. 이러한 스냅샷은 특정 시점의 복사본이므로 시스템 리소스와 공간을 적게 차지합니다.
- 작업할 때 `ontap-nas` 그리고 `ontap-san` 드라이버, ONTAP 스냅샷은 FlexVol 의 특정 시점 복사본이며 FlexVol 자체의 공간을 차지합니다. 이로 인해 스냅샷이 생성되거나 예약됨에 따라 볼륨에서 쓸 수 있는 공간의 양이 시간이 지남에 따라 줄어들 수 있습니다. 이 문제를 해결하는 간단한 방법 중 하나는 Kubernetes를 통해 볼륨 크기를 조정하여 볼륨을 늘리는 것입니다. 또 다른 옵션은 더 이상 필요하지 않은 스냅샷을 삭제하는 것입니다. Kubernetes를 통해 생성된 VolumeSnapshot이 삭제되면 Trident 연관된 ONTAP 스냅샷을 삭제합니다. Kubernetes를 통해 생성되지 않은 ONTAP 스냅샷도 삭제할 수 있습니다.

Trident 사용하면 VolumeSnapshots를 사용하여 새로운 PV를 만들 수 있습니다. 이러한 스냅샷에서 PV를 만드는 작업은 지원되는 ONTAP 및 CVS 백엔드에 대한 FlexClone 기술을 사용하여 수행됩니다. 스냅샷에서 PV를 생성할 때 백업 볼륨은 스냅샷의 부모 볼륨의 FlexClone 입니다. 그만큼 `solidfire-san` 드라이버는 Element 소프트웨어 볼륨 복제본을 사용하여 스냅샷에서 PV를 생성합니다. 여기서는 Element 스냅샷에서 복제본을 만듭니다.

가상 풀은 Trident 스토리지 백엔드와 Kubernetes 간의 추상화 계층을 제공합니다. `StorageClasses` . 이를 통해 관리자는 백엔드에 독립적인 공통적인 방식으로 위치, 성능 및 보호와 같은 측면을 각 백엔드에 대해 정의할 수 있습니다. `StorageClass` 원하는 기준을 충족하기 위해 사용할 물리적 백엔드, 백엔드 풀 또는 백엔드 유형을 지정합니다.

가상 풀에 대해 알아보세요

스토리지 관리자는 JSON 또는 YAML 정의 파일에서 Trident 백엔드에 가상 풀을 정의할 수 있습니다.



가상 풀 목록 외부에 지정된 모든 측면은 백엔드에 전역적으로 적용되며 모든 가상 풀에 적용되는 반면, 각 가상 풀은 하나 이상의 측면을 개별적으로 지정할 수 있습니다(백엔드 전역 측면을 재정의함).

- 가상 풀을 정의할 때 백엔드 정의에서 기존 가상 풀의 순서를 재정렬하려고 하지 마세요.
- 기존 가상 풀의 속성을 수정하지 않는 것이 좋습니다. 변경 사항을 적용하려면 새로운 가상 풀을 정의해야 합니다.

대부분의 측면은 백엔드별 용어로 지정됩니다. 중요한 점은 측면 값이 백엔드 드라이버 외부에 노출되지 않으며 일치에 사용할 수 없다는 것입니다. `StorageClasses` . 대신 관리자는 각 가상 풀에 대해 하나 이상의 레이블을 정의합니다. 각 레이블은 키:값 쌍이며, 레이블은 고유한 백엔드에서 공통적으로 적용될 수 있습니다. 측면과 마찬가지로 레이블은 풀별로 지정하거나 백엔드 전체에 적용할 수 있습니다. 미리 정의된 이름과 값이 있는 측면과 달리 관리자는 필요에 따라 레이블 키와 값을 정의할 수 있는 전적인 재량권을 갖습니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

가상 풀 레이블은 다음 문자를 사용하여 변경할 수 있습니다.

- 대문자 `A-Z`
- 소문자 `a-z`
- 숫자 `0-9`
- 밑줄 `_`
- 하이픈 `-`

에이 `StorageClass` 선택기 매개변수 내의 레이블을 참조하여 사용할 가상 풀을 식별합니다. 가상 풀 선택기는 다음 연산자를 지원합니다.

연산자	예	풀의 레이블 값은 다음과 같아야 합니다.
`=`	성능=프리미엄	성냥
`!=`	성능!=극단적	일치하지 않음
`in`	(동쪽, 서쪽) 위치	값 집합에 속하다
`notin`	성과 노틴(실버, 브론즈)	값 집합에 포함되지 않음
`<key>`	보호	어떤 가치로도 존재
`!<key>`	!보호	존재하지 않음

볼륨 액세스 그룹

Trident 어떻게 사용하는지 자세히 알아보세요 "볼륨 액세스 그룹" .

> CHAP를 사용하는 경우 이 섹션을 무시하세요. CHAP는 관리를 간소화하고 아래 설명된 확장 제한을 방지하는 데 권장됩니다. 또한, CSI 모드에서 Trident 사용하는 경우 이 섹션을 무시할 수 있습니다. Trident 향상된 CSI 프로비저너로 설치될 경우 CHAP를 사용합니다.

볼륨 액세스 그룹에 대해 알아보세요

Trident 볼륨 액세스 그룹을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어할 수 있습니다. CHAP가 비활성화된 경우 CHAP는 다음과 같은 액세스 그룹을 찾을 것으로 예상합니다. `trident` 구성에서 하나 이상의 액세스 그룹 ID를 지정하지 않는 한.

Trident 구성된 액세스 그룹과 새 볼륨을 연결하지만 액세스 그룹 자체를 생성하거나 관리하지는 않습니다. 액세스 그룹은 스토리지 백엔드가 Trident 에 추가되기 전에 존재해야 하며, 해당 백엔드에서 프로비저닝된 볼륨을 잠재적으로 마운트할 수 있는 Kubernetes 클러스터의 모든 노드에서 iSCSI IQN을 포함해야 합니다. 대부분의 설치에는 클러스터의 모든 작업자 노드가 포함됩니다.

노드가 64개 이상인 Kubernetes 클러스터의 경우 여러 액세스 그룹을 사용해야 합니다. 각 액세스 그룹에는 최대 64개의 IQN이 포함될 수 있으며, 각 볼륨은 4개의 액세스 그룹에 속할 수 있습니다. 최대 4개의 액세스 그룹을 구성하면 최대 256개 노드로 구성된 클러스터의 모든 노드가 모든 볼륨에 액세스할 수 있습니다. 볼륨 액세스 그룹에 대한 최신 제한 사항은 다음을 참조하세요. "여기" .

기본 구성을 사용하는 구성을 수정하는 경우 `trident` 다른 사용자도 사용하는 그룹에 액세스하려면 해당 ID를 포함합니다. `trident` 목록에 있는 그룹에 접근합니다.

Trident 빠른 시작

몇 단계만 거치면 Trident 설치하고 스토리지 리소스 관리를 시작할 수 있습니다. 시작하기 전에 검토하세요"Trident 요구 사항" .

Docker의 경우 다음을 참조하세요."Docker용 Trident" .

1 워커 노드 준비

Kubernetes 클러스터의 모든 워커 노드는 Pod에 대해 프로비저닝한 볼륨을 마운트할 수 있어야 합니다.

"워커 노드 준비"

2 Trident 설치

Trident 다양한 환경과 조직에 최적화된 여러 가지 설치 방법과 모드를 제공합니다.

"Trident 설치"

3 백엔드 만들기

백엔드는 Trident 와 스토리지 시스템 간의 관계를 정의합니다. 이는 Trident 해당 스토리지 시스템과 통신하는 방법과 Trident 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

"백엔드 구성"귀하의 저장 시스템을 위해

4 Kubernetes StorageClass 만들기

Kubernetes StorageClass 객체는 Trident 프로비저너로 지정하고 사용자 정의 가능한 속성으로 볼륨을 프로비저닝하는 스토리지 클래스를 생성할 수 있도록 합니다. Trident Trident 프로비저너를 지정하는 Kubernetes 객체에 맞는 스토리지 클래스를 생성합니다.

"스토리지 클래스 생성"

5 볼륨 제공

PersistentVolume (PV)은 Kubernetes 클러스터에서 클러스터 관리자가 프로비저닝하는 물리적 스토리지 리소스입니다. *PersistentVolumeClaim* (PVC)은 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

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

"볼륨 제공"

다음은 무엇인가요?

이제 추가 백엔드를 추가하고, 스토리지 클래스를 관리하고, 백엔드를 관리하고, 볼륨 작업을 수행할 수 있습니다.

요구 사항

Trident 설치하기 전에 다음과 같은 일반적인 시스템 요구 사항을 검토하세요. 특정 백엔드에는 추가 요구 사항이 있을 수 있습니다.

Trident 에 대한 중요 정보

- Trident 에 대한 다음의 중요 정보를 꼭 읽어보세요.*

** Trident** 에 대한 중요 정보****

- Kubernetes 1.34가 이제 Trident 에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident 업그레이드하세요.
- Trident SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며 권장 값은 다음과 같습니다. `find_multipaths: no` multipath.conf 파일에서.

  비다중경로 구성 사용 또는 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트가 실패합니다. Trident 다음을 사용할 것을 권장했습니다. `find_multipaths: no` 21.07 릴리스 이후.

지원되는 프런트엔드(오케스트레이터)

Trident 다음을 포함하여 다양한 컨테이너 엔진과 오케스트레이터를 지원합니다.

- Anthos On-Prem(VMware) 및 Anthos on bare metal 1.16
- 쿠버네티스 1.27 - 1.34
- OpenShift 4.12, 4.14 - 4.19(OpenShift 4.19에서 iSCSI 노드 준비를 사용할 계획인 경우 지원되는 최소 Trident 버전은 25.06.1입니다.)

> Trident 다음과 같은 기준으로 이전 OpenShift 버전을 계속 지원합니다."Red Hat Extended Update Support(EUS) 릴리스 수명 주기" 더 이상 공식적으로 지원되지 않는 Kubernetes 버전을 사용하는 경우에도 마찬가지입니다. 이런 경우 Trident 설치하는 동안 Kubernetes 버전에 대한 경고 메시지는 무시해도 됩니다.

- Rancher Kubernetes Engine 2(RKE2) v1.27.x - 1.34.x

> Trident 는 Rancher Kubernetes Engine 2(RKE2) 버전 1.27.x - 1.34.x에서 지원되지만, Trident 현재 RKE2 v1.28.5+rke2r1에서만 검증되었습니다._

Trident Google Kubernetes Engine(GKE), Amazon Elastic Kubernetes Services(EKS), Azure Kubernetes

Service(AKS), Mirantis Kubernetes Engine(MKE), VMWare Tanzu Portfolio를 포함한 다양한 완전 관리형 및 자체 관리형 Kubernetes 제품과도 호환됩니다.

Trident 와 ONTAP 스토리지 공급자로 사용될 수 있습니다."큐브비트" .

> Trident 설치된 Kubernetes 클러스터를 1.25에서 1.26 이상으로 업그레이드하기 전에 다음을 참조하세요."Helm 설치 업그레이드" .

지원되는 백엔드(스토리지)

Trident 사용하려면 다음 지원되는 백엔드 중 하나 이상이 필요합니다.

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN 어레이(ASA)
- NetApp의 제한된 지원 하에 온프레미스 FAS, AFF, Select 또는 ASA r2(iSCSI 및 NVMe/TCP) 클러스터 버전이 제공됩니다. 보다"소프트웨어 버전 지원" .
- NetApp HCI/Element 소프트웨어 11 이상

KubeVirt 및 OpenShift 가상화에 대한 Trident 지원

지원되는 스토리지 드라이버:

Trident KubeVirt 및 OpenShift Virtualization에 대해 다음 ONTAP 드라이버를 지원합니다.

- 온탑나스
- 온탑-나스-이코노미
- ontap-san(iSCSI, FCP, NVMe over TCP)
- ontap-san-economy(iSCSI 전용)

고려해야 할 사항:

- 저장 클래스를 업데이트하여 다음을 수행하세요. `fsType` 매개변수(예: `fsType: "ext4"`) OpenShift 가상화 환경에서. 필요한 경우 볼륨 모드를 명시적으로 차단으로 설정하십시오. `volumeMode=Block` 매개변수 `dataVolumeTemplates` CDI에 블록 데이터 볼륨을 생성하도록 알립니다.
- 블록 스토리지 드라이버를 위한 *RWX* 액세스 모드: ontap-san(iSCSI, NVMe/TCP, FC) 및 ontap-san-economy(iSCSI) 드라이버는 "volumeMode: Block"(원시 장치)에서만 지원됩니다. 이러한 운전자의 경우 `fstype` 볼륨이 원시 장치 모드로 제공되므로 매개변수를 사용할 수 없습니다.
- RWX 액세스 모드가 필요한 라이브 마이그레이션 워크플로의 경우 다음 조합이 지원됩니다.
  - NFS+ `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (원시 장치)
  - NVMe/TCP + `volumeMode=Block` (원시 장치)
  - FC + `volumeMode=Block` (원시 장치)

기능 요구 사항

아래 표는 이번 Trident 릴리스에서 사용할 수 있는 기능과 이를 지원하는 Kubernetes 버전을 요약한 것입니다.

특징	쿠버네티스 버전	기능 게이트가 필요합니까?
Trident	1.27 - 1.34	아니요
볼륨 스냅샷	1.27 - 1.34	아니요
볼륨 스냅샷의 PVC	1.27 - 1.34	아니요
iSCSI PV 크기 조정	1.27 - 1.34	아니요
ONTAP 양방향 CHAP	1.27 - 1.34	아니요
동적 수출 정책	1.27 - 1.34	아니요
Trident 오퍼레이터	1.27 - 1.34	아니요
CSI 토폴로지	1.27 - 1.34	아니요

테스트된 호스트 운영 체제

Trident 공식적으로 특정 운영 체제를 지원하지 않지만, 다음은 작동하는 것으로 알려져 있습니다.

- AMD64 및 ARM64 환경에서 OpenShift 컨테이너 플랫폼이 지원하는 Red Hat Enterprise Linux CoreOS(RHCOS) 버전
- AMD64 및 ARM64 기반 Red Hat Enterprise Linux(RHEL) 8 이상

  NVMe/TCP에는 RHEL 9 이상이 필요합니다.

- AMD64 및 ARM64에서 Ubuntu 22.04 LTS 이상 버전
- 윈도우 서버 2022
- SUSE Linux Enterprise Server (SLES) 15 이상

기본적으로 Trident 컨테이너에서 실행되므로 모든 Linux 워커에서 실행됩니다. 하지만 해당 작업자는 사용하는 백엔드에 따라 표준 NFS 클라이언트나 iSCSI 이니시에이터를 사용하여 Trident 제공하는 볼륨을 마운트할 수 있어야 합니다.

그만큼 `tridentctl` 이 유틸리티는 모든 Linux 배포판에서도 실행됩니다.

호스트 구성

Kubernetes 클러스터의 모든 워커 노드는 Pod에 대해 프로비저닝한 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS, iSCSI 또는 NVMe 도구를 설치해야 합니다.

"워커 노드 준비"

스토리지 시스템 구성

Trident 백엔드 구성에서 사용하려면 먼저 스토리지 시스템을 변경해야 할 수도 있습니다.

"백엔드 구성"

Trident 포트

Trident 통신을 위해 특정 포트에 접근해야 합니다.

"Trident 포트"

컨테이너 이미지 및 해당 Kubernetes 버전

공기 간격 설치의 경우, 다음 목록은 Trident 설치하는 데 필요한 컨테이너 이미지의 참고 자료입니다. 사용하다 `tridentctl images` 필요한 컨테이너 이미지 목록을 확인하는 명령입니다.

Trident 25.06.2에 필요한 컨테이너 이미지

쿠버네티스 버전	컨테이너 이미지
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0	• docker.io/netapp/trident:25.06.2 • docker.io/netapp/trident-autosupport:25.06 • registry.k8s.io/sig-storage/csi-provisioner:v5.2.0 • registry.k8s.io/sig-storage/csi-attacher:v4.8.1 • registry.k8s.io/sig-storage/csi-resizer:v1.13.2 • registry.k8s.io/sig-storage/csi-snapshotter:v8.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.13.0 • docker.io/netapp/trident-operator:25.06.2 (선택 사항)

Trident 25.06에 필요한 컨테이너 이미지

쿠버네티스 버전	컨테이너 이미지
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0	• docker.io/netapp/trident:25.06.0 • docker.io/netapp/trident-autosupport:25.06 • registry.k8s.io/sig-storage/csi-provisioner:v5.2.0 • registry.k8s.io/sig-storage/csi-attacher:v4.8.1 • registry.k8s.io/sig-storage/csi-resizer:v1.13.2 • registry.k8s.io/sig-storage/csi-snapshotter:v8.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.13.0 • docker.io/netapp/trident-operator:25.06.0 (선택 사항)

Trident 설치

Trident 연산자를 사용하여 설치

tridentctl을 사용하여 설치

OpenShift 인증 운영자를 사용하여 설치하세요

Trident 사용

워커 노드 준비

Kubernetes 클러스터의 모든 워커 노드는 Pod에 대해 프로비저닝한 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS, iSCSI, NVMe/TCP 또는 FC 도구를 설치해야 합니다.

올바른 도구 선택

여러 드라이버를 조합하여 사용하는 경우 드라이버에 필요한 모든 도구를 설치해야 합니다. Red Hat Enterprise Linux CoreOS(RHCOS)의 최신 버전에는 기본적으로 도구가 설치되어 있습니다.

NFS 도구

"NFS 도구 설치"다음을 사용하는 경우: `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `azure-netapp-files` , `gcp-cvs` .

iSCSI 도구

"iSCSI 도구 설치"다음을 사용하는 경우: `ontap-san` , `ontap-san-economy` , `solidfire-san` .

NVMe 도구

"NVMe 도구 설치"당신이 사용하는 경우 `ontap-san` TCP(NVMe/TCP) 프로토콜을 통한 비휘발성 메모리 익스프레스(NVMe)용입니다.

NetApp NVMe/TCP에 ONTAP 9.12 이상을 권장합니다.

FC 도구를 통한 **SCSI**

참조하다"FC 및 FC-NVMe SAN 호스트를 구성하는 방법" FC 및 FC-NVMe SAN 호스트 구성에 대한 자세한 내용은 다음을 참조하세요.

"FC 도구 설치"당신이 사용하는 경우 `ontap-san` sanType으로 `fcp` (FC를 통한 SCSI).

고려 사항: * FC를 통한 SCSI는 OpenShift 및 KubeVirt 환경에서 지원됩니다. * Docker에서는 FC를 통한 SCSI가 지원되지 않습니다. * iSCSI 자체 복구 기능은 FC를 통한 SCSI에는 적용되지 않습니다.

노드 서비스 검색

Trident 노드가 iSCSI 또는 NFS 서비스를 실행할 수 있는지 자동으로 감지하려고 시도합니다.

노드 서비스 검색은 검색된 서비스를 식별하지만 서비스가 올바르게 구성되었는지 보장하지는 않습니다. 반대로, 검색된 서비스가 없다고 해서 볼륨 마운트가 실패한다는 보장은 없습니다.

이벤트 검토

Trident 노드가 발견된 서비스를 식별할 수 있도록 이벤트를 생성합니다. 이러한 이벤트를 검토하려면 다음을 실행하세요.

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

발견된 서비스 검토

Trident Trident 노드 CR의 각 노드에서 활성화된 서비스를 식별합니다. 검색된 서비스를 보려면 다음을 실행하세요.

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS 볼륨

운영 체제에 맞는 명령을 사용하여 NFS 도구를 설치합니다. 부팅 시 NFS 서비스가 시작되었는지 확인하세요.

RHEL 8 이상

```
sudo yum install -y nfs-utils
```

우분투

```
sudo apt-get install -y nfs-common
```

컨테이너에 볼륨을 연결할 때 오류가 발생하는 것을 방지하려면 NFS 도구를 설치한 후 작업자 노드를 재부팅하세요.

iSCSI 볼륨

Trident 자동으로 iSCSI 세션을 설정하고, LUN을 스캔하고, 다중 경로 장치를 검색하여 포맷하고, 포드에 마운트할 수 있습니다.

iSCSI 자체 복구 기능

ONTAP 시스템의 경우 Trident 다음을 위해 5분마다 iSCSI 자체 복구를 실행합니다.

1. 원하는 iSCSI 세션 상태와 현재 iSCSI 세션 상태를 *식별*합니다.
2. 원하는 상태와 현재 상태를 *비교*하여 필요한 수리를 파악합니다. Trident 수리 우선순위를 결정하고 언제 수리를 먼저 시작해야 할지 결정합니다.
3. 현재 iSCSI 세션 상태를 원하는 iSCSI 세션 상태로 되돌리려면 필요한 수리를 수행합니다.

자체 복구 활동 로그는 다음 위치에 있습니다. `trident-main` 해당 Daemonset 포드의 컨테이너입니다. 로그를 보려면 다음을 설정해야 합니다. `debug` Trident 설치 중에 "true"로 설정합니다.

Trident iSCSI 자체 복구 기능은 다음을 방지하는 데 도움이 될 수 있습니다.

- 네트워크 연결 문제 이후 발생할 수 있는 오래되거나 비정상적 iSCSI 세션입니다. 세션이 오래된 경우, Trident 포털과의 연결을 재설정하기 위해 로그아웃하기 전에 7분을 기다립니다.

> 예를 들어, CHAP 비밀번호가 스토리지 컨트롤러에서 순환되고 네트워크 연결이 끊어지면 이전(*stale*) CHAP 비밀번호가 유지될 수 있습니다. 자가 복구 기능은 이를 인식하고 업데이트된 CHAP 비밀을 적용하기 위해 세션을 자동으로 재설정합니다.

- iSCSI 세션이 누락되었습니다
- LUN이 누락되었습니다
- Trident 업그레이드 전 고려사항*
- 노드당 igroup(23.04+에서 도입)만 사용 중인 경우 iSCSI 자체 복구 기능은 SCSI 버스의 모든 장치에 대한 SCSI 재검색을 시작합니다.
- 백엔드 범위의 igroup(23.04부터 더 이상 사용되지 않음)만 사용 중인 경우 iSCSI 자체 복구 기능은 SCSI 버스에서 정확한 LUN ID를 찾기 위해 SCSI 재검색을 시작합니다.
- 노드별 igroup과 백엔드 범위 igroup을 혼합하여 사용하는 경우 iSCSI 자체 복구 기능은 SCSI 버스에서 정확한 LUN ID를 찾기 위해 SCSI 재검색을 시작합니다.

iSCSI 도구 설치

운영 체제에 맞는 명령을 사용하여 iSCSI 도구를 설치합니다.

시작하기 전에

- Kubernetes 클러스터의 각 노드에는 고유한 IQN이 있어야 합니다. 이것은 필수 전제 조건입니다.
- RHCOS 버전 4.5 이상 또는 기타 RHEL 호환 Linux 배포판을 사용하는 경우 `solidfire-san` 드라이버 및 Element OS 12.5 이하에서는 CHAP 인증 알고리즘이 MD5로 설정되어 있는지 확인하십시오. `/etc/iscsi/iscsid.conf` Element 12.7에서는 보안 FIPS 호환 CHAP 알고리즘 SHA1, SHA-256 및 SHA3-256을 사용할 수 있습니다.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- iSCSI PV와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 다음을 지정합니다. `discard` StorageClass의 mountOption을 사용하여 인라인 공간 회수를 수행합니다. 참조하다 "Red Hat 문서" .
- 최신 버전으로 업그레이드했는지 확인하세요. `multipath-tools` .

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. iscsi-initiator-utils 버전이 6.2.0.874-2.el7 이상인지 확인하세요.

```
rpm -q iscsi-initiator-utils
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `iscsid` 그리고 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now iscsid multipathd
```

6. 활성화하고 시작하세요 `iscsi` :

```
sudo systemctl enable --now iscsi
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic의 경우) 또는 2.0.874-7.1ubuntu6.1 이상(focal의 경우)인지 확인하세요.

```
dpkg -l open-iscsi
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `open-iscsi` 그리고 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Ubuntu 18.04의 경우 대상 포트를 검색해야 합니다. `iscsiadm` 시작하기 전에 `open-iscsi` iSCSI 데몬을 시작하려면. 또는 다음을 수정할 수 있습니다. `iscsi` 서비스를 시작하려면 `iscsid` 자동으로.

iSCSI 자체 복구 구성 또는 비활성화

다음의 Trident iSCSI 자체 복구 설정을 구성하여 오래된 세션을 수정할 수 있습니다.

- **iSCSI** 자체 복구 간격: iSCSI 자체 복구가 호출되는 빈도를 결정합니다(기본값: 5분). 작은 숫자를 설정하면 더 자주 실행되도록 구성할 수 있고, 큰 숫자를 설정하면 덜 자주 실행되도록 구성할 수 있습니다.

> iSCSI 자체 복구 간격을 0으로 설정하면 iSCSI 자체 복구가 완전히 중지됩니다. iSCSI 자체 복구를 비활성화하는 것은 권장하지 않습니다. iSCSI 자체 복구가 의도한 대로 작동하지 않거나 디버깅 목적으로만 특정 상황에서만 비활성화해야 합니다.

- **iSCSI** 자체 복구 대기 시간: iSCSI 자체 복구가 비정상 세션에서 로그아웃하고 다시 로그인을 시도하기 전까지 기다리는 시간을 결정합니다(기본값: 7분). 세션이 비정상적으로 식별되어 로그아웃되기 전에 더 오래 기다린 후 다시 로그인을 시도하도록 하려면 숫자를 더 크게 구성하거나, 로그아웃한 후 더 일찍 로그인하도록 숫자를 더 작게 구성할 수 있습니다.

지배

iSCSI 자체 복구 설정을 구성하거나 변경하려면 다음을 전달하세요. `iscsiSelfHealingInterval` 그리고 `iscsiSelfHealingWaitTime` Helm 설치 또는 Helm 업데이트 중의 매개변수.

다음 예에서는 iSCSI 자체 복구 간격을 3분으로 설정하고 자체 복구 대기 시간을 6분으로 설정합니다.

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

트라이던트**ctl**

iSCSI 자체 복구 설정을 구성하거나 변경하려면 다음을 전달하세요. `iscsi-self-healing-interval` 그리고 `iscsi-self-healing-wait-time` tridentctl 설치 또는 업데이트 중의 매개변수.

다음 예에서는 iSCSI 자체 복구 간격을 3분으로 설정하고 자체 복구 대기 시간을 6분으로 설정합니다.

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCP 볼륨

운영 체제에 맞는 명령을 사용하여 NVMe 도구를 설치하세요.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 커널 버전에서 NVMe 패키지를 사용할 수 없는 경우, 노드의 커널 버전을 NVMe 패키지가 있는 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

우분투

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

설치 확인

설치 후 다음 명령을 사용하여 Kubernetes 클러스터의 각 노드에 고유한 NQN이 있는지 확인하세요.

```
cat /etc/nvme/hostnqn
```

Trident 다음을 수정합니다. `ctrl_device_tmo` NVMe가 다운되더라도 경로를 포기하지 않도록 보장하는 가치입니다. 이 설정을 변경하지 마세요.

FC 볼륨을 통한 SCSI

이제 Trident 와 함께 Fibre Channel(FC) 프로토콜을 사용하여 ONTAP 시스템에서 스토리지 리소스를 프로비저닝하고 관리할 수 있습니다.

필수 조건

FC에 필요한 네트워크 및 노드 설정을 구성합니다.

네트워크 설정

1. 대상 인터페이스의 WWPN을 가져옵니다. 참조하다 "네트워크 인터페이스 표시" 자세한 내용은.
2. 개시자(호스트)의 인터페이스에 대한 WWPN을 가져옵니다.

   해당 호스트 운영 체제 유틸리티를 참조하세요.
3. 호스트와 대상의 WWPN을 사용하여 FC 스위치에서 구역화를 구성합니다.

   자세한 내용은 해당 스위치 공급업체의 설명서를 참조하세요.

   자세한 내용은 다음 ONTAP 문서를 참조하세요.

   - "파이버 채널 및 FCoE 구역화 개요"

- "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"

FC 도구 설치

운영 체제에 맞는 명령을 사용하여 FC 도구를 설치하세요.

- FC PV와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 다음을 지정합니다. `discard` StorageClass의 mountOption을 사용하여 인라인 공간 회수를 수행합니다. 참조하다 "Red Hat 문서" .

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now multipathd
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> 보장하다 `/etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
```

백엔드 구성 및 관리

백엔드 구성

백엔드는 Trident 와 스토리지 시스템 간의 관계를 정의합니다. 이는 Trident 해당 스토리지 시스템과 통신하는 방법과 Trident 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

Trident 스토리지 클래스에서 정의한 요구 사항에 맞는 백엔드의 스토리지 풀을 자동으로 제공합니다. 스토리지 시스템의 백엔드를 구성하는 방법을 알아보세요.

- "Azure NetApp Files 백엔드 구성"
- "Google Cloud NetApp Volumes 백엔드 구성"
- "Google Cloud Platform 백엔드에 Cloud Volumes Service 구성"
- "NetApp HCI 또는 SolidFire 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버를 사용하여 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버를 사용하여 백엔드 구성"
- "Amazon FSx for NetApp ONTAP 과 함께 Trident 사용"

Azure NetApp Files

Azure NetApp Files 백엔드 구성

Azure NetApp Files Trident 의 백엔드로 구성할 수 있습니다. Azure NetApp Files 백엔드를 사용하여 NFS 및 SMB 볼륨을 연결할 수 있습니다. Trident 또한 Azure Kubernetes Services(AKS) 클러스터에 대한 관리형 ID를 사용하여 자격 증명 관리를 지원합니다.

Azure NetApp Files 드라이버 세부 정보

Trident 클러스터와 통신하기 위해 다음과 같은 Azure NetApp Files 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`azure-netapp-files`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

고려 사항

- Azure NetApp Files 서비스는 50GiB보다 작은 볼륨을 지원하지 않습니다. 더 작은 볼륨이 요청되면 Trident 자동으로 50GiB 볼륨을 생성합니다.
- Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.

AKS의 관리 ID

Trident 지지대"관리되는 ID" Azure Kubernetes Services 클러스터용. 관리형 ID가 제공하는 간소화된 자격 증명 관리를 활용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 관리 ID
- 다음을 포함하는 Trident 설치됨 `cloudProvider` 지정하다 `"Azure"` .

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 `"Azure"` . 예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

지배

다음 예제에서는 Trident 세트를 설치합니다. `cloudProvider` 환경 변수를 사용하여 Azure에 `$CP` :

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>트라이던트ctl</code>

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 플래그를 `Azure` :

```
tridentctl install --cloud-provider="Azure" -n trident
```

AKS용 클라우드 ID

클라우드 ID를 사용하면 Kubernetes Pod가 명시적인 Azure 자격 증명을 제공하는 대신 워크로드 ID로 인증하여 Azure 리소스에 액세스할 수 있습니다.

Azure에서 클라우드 ID를 활용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터

- AKS Kubernetes 클러스터에 구성된 워크로드 ID 및 oidc-issuer
- 다음을 포함하는 Trident 설치됨 `cloudProvider` 지정하다 `"Azure"` 그리고 `cloudIdentity` 워크로드 ID 지정

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 `"Azure"` 그리고 설정하다 `cloudIdentity` 에게 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` .

예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

지배

다음 환경 변수를 사용하여 **cloud-provider (CP)** 및 **cloud-identity (CI)** 플래그의 값을 설정합니다.

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 환경 변수를 사용하여 Azure에 `$CP` 그리고 설정합니다 `cloudIdentity` 환경 변수 사용 `$CI` :

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>트라이던트ctl</code>

다음 환경 변수를 사용하여 클라우드 공급자 및 클라우드 **ID** 플래그의 값을 설정합니다.

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloud-provider` 플래그를 `$CP` , 그리고 `cloud-identity` 에게 `$CI` :

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Azure NetApp Files 백엔드 구성을 준비합니다.

Azure NetApp Files 백엔드를 구성하기 전에 다음 요구 사항이 충족되는지 확인해야 합니다.

NFS 및 **SMB** 볼륨에 대한 필수 구성 요소

Azure NetApp Files 처음 사용하거나 새로운 위치에서 사용하는 경우 Azure NetApp Files를 설정하고 NFS 볼륨을 생성하기 위해 일부 초기 구성이 필요합니다. 참조하다 "Azure: Azure NetApp Files 설정 및 NFS 볼륨 생성" .

구성하고 사용하려면 "Azure NetApp Files" 백엔드에는 다음이 필요합니다.

- `subscriptionID`, `tenantID` , `clientID` , `location` , 그리고 `clientSecret` AKS 클러스터에서 관리형 ID를 사용하는 경우 선택 사항입니다.
- `tenantID`, `clientID` , 그리고 `clientSecret` AKS 클러스터에서 클라우드 ID를 사용하는 경우 선택 사항입니다.

- 수용 인원 풀. 참조하다"Microsoft: Azure NetApp Files 대한 용량 풀 만들기" .
- Azure NetApp Files 에 위임된 서브넷입니다. 참조하다"Microsoft: Azure NetApp Files 에 서브넷 위임" .
- `subscriptionID`Azure NetApp Files 활성화된 Azure 구독에서.
- `tenantID`, `clientID` , 그리고 `clientSecret` 에서"앱 등록" Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory에 있습니다. 앱 등록에는 다음 중 하나를 사용해야 합니다.
  - 소유자 또는 기여자 역할"Azure에서 미리 정의됨" .
  - 에이"사용자 정의 기여자 역할" 구독 수준에서(`assignableScopes`) Trident 에 필요한 것으로만 제한된 다음 권한이 있습니다. 사용자 정의 역할을 만든 후,"Azure Portal을 사용하여 역할 할당" .

사용자 정의 기여자 역할

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/delete",
          "Microsoft.Features/features/read",
          "Microsoft.Features/operations/read",
          "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

- Azure `location` 적어도 하나 이상 포함 "위임된 서브넷" . Trident 22.01부터 `location` 매개변수는 백엔드 구성 파일의 최상위에 있는 필수 필드입니다. 가상 풀에 지정된 위치 값은 무시됩니다.
- 사용하려면 `Cloud Identity` , 얻으세요 `client ID` 에서 "사용자가 할당한 관리 ID" 그리고 해당 ID를 지정하세요 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` .

SMB 볼륨에 대한 추가 요구 사항

SMB 볼륨을 생성하려면 다음이 필요합니다.

- Active Directory가 구성되고 Azure NetApp Files 에 연결되었습니다. 참조하다"Microsoft: Azure NetApp Files 대한 Active Directory 연결 만들기 및 관리" .
- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Azure NetApp Files Active Directory에 인증할 수 있도록 Active Directory 자격 증명이 포함된 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

Azure NetApp Files 백엔드 구성 옵션 및 예제

Azure NetApp Files 에 대한 NFS 및 SMB 백엔드 구성 옵션에 대해 알아보고 구성 예를

검토하세요.

백엔드 구성 옵션

Trident 백엔드 구성(서브넷, 가상 네트워크, 서비스 수준 및 위치)을 사용하여 요청된 위치에서 사용 가능하고 요청된 서비스 수준 및 서브넷과 일치하는 용량 풀에 Azure NetApp Files 볼륨을 만듭니다.

* NetApp Trident 25.06 릴리스부터 수동 QoS 용량 풀이 기술 미리 보기로 지원됩니다.*

Azure NetApp Files 백엔드는 다음과 같은 구성 옵션을 제공합니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	"azure-netapp-파일"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	운전자 이름 + "_" + 임의의 문자
`subscriptionID`	AKS 클러스터에서 관리 ID가 활성화된 경우 Azure 구독의 구독 ID는 선택 사항입니다.	
`tenantID`	AKS 클러스터에서 관리 ID 또는 클라우드 ID가 사용되는 경우 앱 등록 선택 사항의 테넌트 ID입니다.	
`clientID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID가 사용되는 경우 앱 등록 선택 사항의 클라이언트 ID입니다.	
`clientSecret`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록 선택 사항의 클라이언트 비밀번호입니다.	
`serviceLevel`	중 하나 `Standard` , `Premium` , 또는 `Ultra`	"" (무작위의)
`location`	새 볼륨이 생성될 Azure 위치의 이름입니다. AKS 클러스터에서 관리 ID가 활성화된 경우 선택 사항입니다.	
`resourceGroups`	검색된 리소스를 필터링하기 위한 리소스 그룹 목록	"[]" (필터 없음)
`netappAccounts`	검색된 리소스를 필터링하기 위한 NetApp 계정 목록	"[]" (필터 없음)
`capacityPools`	검색된 리소스를 필터링하기 위한 용량 풀 목록	"[]" (필터 없음, 무작위)
`virtualNetwork`	위임된 서브넷이 있는 가상 네트워크의 이름	""
`subnet`	위임된 서브넷의 이름 `Microsoft.Netapp/volumes`	""

매개변수	설명	기본
`networkFeatures`	볼륨에 대한 VNet 기능 세트는 다음과 같습니다. `Basic` 또는 `Standard` . 네트워크 기능은 일부 지역에서는 제공되지 않으며 구독을 통해 활성화해야 할 수도 있습니다. 지정 `networkFeatures` 해당 기능이 활성화되지 않으면 볼륨 프로비저닝이 실패합니다.	""
`nfsMountOptions`	NFS 마운트 옵션에 대한 세부적인 제어. SMB 볼륨에서는 무시됩니다. NFS 버전 4.1을 사용하여 볼륨을 마운트하려면 다음을 포함합니다. `nfsvers=4` 쉼표로 구분된 마운트 옵션 목록에서 NFS v4.1을 선택하세요. 스토리지 클래스 정의에 설정된 마운트 옵션은 백엔드 구성에 설정된 마운트 옵션보다 우선합니다.	"nfsvers=3"
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, `\{"api": false, "method": true, "discovery": true}` . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 `nfs` , `smb` 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .	
`qosType`	QoS 유형(자동 또는 수동)을 나타냅니다. * Trident 25.06 기술 미리보기*	자동
`maxThroughput`	허용되는 최대 처리량을 MiB/초 단위로 설정합니다. 수동 QoS 용량 풀에서만 지원됩니다. * Trident 25.06 기술 미리보기*	`4 MiB/sec`

네트워크 기능에 대한 자세한 내용은 다음을 참조하세요."Azure NetApp Files 볼륨에 대한 네트워크 기능 구성" .

필요한 권한 및 리소스

PVC를 생성할 때 "용량 풀을 찾을 수 없습니다" 오류가 발생하는 경우 앱 등록에 필요한 권한과 리소스(서브넷, 가상 네트워크, 용량 풀)가 연결되어 있지 않을 가능성이 높습니다. 디버그가 활성화된 경우 Trident 백엔드가 생성될 때 검색된 Azure 리소스를 기록합니다. 적절한 역할이 사용되고 있는지 확인하세요.

에 대한 값 `resourceGroups` , `netappAccounts` , `capacityPools` , `virtualNetwork` , 그리고 `subnet` 짧은 이름이나 완전한 이름을 사용하여 지정할 수 있습니다. 대부분의 상황에서는 완전한 이름을 사용하는 것이 좋습니다. 짧은 이름은 동일한 이름을 가진 여러 리소스와 일치할 수 있기 때문입니다.

그만큼 `resourceGroups` , `netappAccounts` , 그리고 `capacityPools` 값은 검색된 리소스 세트를 이 스토리지 백엔드에서 사용 가능한 리소스로 제한하는 필터이며, 원하는 대로 조합하여 지정할 수 있습니다. 완전히 정의된 이름은 다음 형식을 따릅니다.

유형	체재
리소스 그룹	<리소스 그룹>
NetApp 계정	<리소스 그룹>/<netapp 계정>
용량 풀	<리소스 그룹>/<netapp 계정>/<용량 풀>
가상 네트워크	<리소스 그룹>/<가상 네트워크>
서브넷	<리소스 그룹>/<가상 네트워크>/<서브넷>

볼륨 프로비저닝

구성 파일의 특정 섹션에서 다음 옵션을 지정하여 기본 볼륨 프로비저닝을 제어할 수 있습니다. 참조하다 구성 예 자세한 내용은.

매개변수	설명	기본
`exportRule`	새로운 볼륨에 대한 내보내기 규칙. `exportRule` CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷의 조합을 쉼표로 구분한 목록이어야 합니다. SMB 볼륨에서는 무시됩니다.	"0.0.0.0/0"
`snapshotDir`	.snapshot 디렉토리의 가시성을 제어합니다.	NFSv4의 경우 "true", NFSv3의 경우 "false"
`size`	새 볼륨의 기본 크기	"100G"
`unixPermissions`	새로운 볼륨의 유닉스 권한(8진수 4자리). SMB 볼륨에서는 무시됩니다.	""(미리보기 기능, 구독 시 허용 목록에 추가 필요)

구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 백엔드의 최소 구성입니다. 이 구성을 사용하면 Trident 구성된 위치에서 Azure NetApp Files 에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀 및 서브넷 중 하나에 무작위로 새 볼륨을 배치합니다. 왜냐하면 `nasType` 생략됩니다. `nfs` 기본값이 적용되고 백엔드가 NFS 볼륨을 프로비저닝합니다.

이 구성은 Azure NetApp Files 처음 시작하고 여러 가지를 시도해 볼 때 이상적이지만 실제로는 프로비저닝하는 볼륨에 대한 추가 범위를 제공해야 합니다.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

AKS의 관리 ID

이 백엔드 구성에서는 다음을 생략합니다. `subscriptionID` , `tenantID` , `clientID` , 그리고 `clientSecret` 관리형 ID를 사용할 때는 선택 사항입니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

AKS용 클라우드 ID

이 백엔드 구성에서는 다음을 생략합니다. `tenantID` , `clientID` , 그리고 `clientSecret` 클라우드 ID를 사용할 때는 선택 사항입니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

용량 풀 필터를 사용한 특정 서비스 수준 구성

이 백엔드 구성은 Azure의 볼륨을 배치합니다. `eastus` 위치 `Ultra` 용량 풀. Trident 해당 위치에서 Azure NetApp Files 에 위임된 모든 서브넷을 자동으로 검색하고 그 중 하나에 무작위로 새 볼륨을 배치합니다.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

수동 **QoS** 용량 풀을 사용한 백엔드 예제

이 백엔드 구성은 Azure의 볼륨을 배치합니다. `eastus` 수동 QoS 용량 풀이 있는 위치입니다. * NetApp Trident 25.06의 기술 미리보기*.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

고급 구성

이 백엔드 구성은 볼륨 배치 범위를 단일 서브넷으로 더욱 줄이고 일부 볼륨 프로비저닝 기본값도 수정합니다.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

이 백엔드 구성은 단일 파일에 여러 개의 스토리지 풀을 정의합니다. 이 기능은 다양한 서비스 수준을 지원하는 여러 용량 풀이 있고 이를 나타내는 Kubernetes의 스토리지 클래스를 생성하려는 경우에 유용합니다. 가상 풀 레이블은 다음을 기준으로 풀을 구별하는 데 사용되었습니다. `performance` .

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - ultra-1
      - ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - standard-1
      - standard-2
```

지원되는 토폴로지 구성

Trident 지역 및 가용성 영역에 따라 워크로드에 대한 볼륨 프로비저닝을 용이하게 합니다. 그만큼 `supportedTopologies` 이 백엔드 구성의 블록은 백엔드당 지역 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 지역 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 지역 및 영역 값과 일치해야 합니다. 이러한 지역과 영역은 저장 클래스에서 제공될 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공하는 지역 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우, Trident 언급된 지역 및 영역에 볼륨을 생성합니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

스토리지 클래스 정의

다음 `StorageClass` 정의는 위의 저장 풀을 참조합니다.

다음을 사용한 정의 예 `parameter.selector` 필드

사용 중 `parameter.selector` 각각에 대해 지정할 수 있습니다 `StorageClass` 볼륨을 호스팅하는 데 사용되는 가상 풀입니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

SMB 볼륨에 대한 예제 정의

사용 중 `nasType` , `node-stage-secret-name` , 그리고 `node-stage-secret-namespace` SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`SMB 볼륨을 지원하는 풀에 대한 필터입니다. `nasType: nfs 또는 nasType: null NFS 풀에 대한 필터.

백엔드 만들기

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes 백엔드 구성

이제 Google Cloud NetApp Volumes Trident 의 백엔드로 구성할 수 있습니다. Google Cloud NetApp Volumes 백엔드를 사용하여 NFS 및 SMB 볼륨을 연결할 수 있습니다.

Google Cloud NetApp Volumes 드라이버 세부 정보

Trident 다음을 제공합니다. `google-cloud-netapp-volumes` 클러스터와 통신하는 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`google-cloud-netapp-volumes`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

GKE용 클라우드 ID

클라우드 ID를 사용하면 Kubernetes 포드가 명시적인 Google Cloud 자격 증명을 제공하는 대신 워크로드 ID로 인증하여 Google Cloud 리소스에 액세스할 수 있습니다.

Google Cloud에서 클라우드 ID를 활용하려면 다음이 필요합니다.

- GKE를 사용하여 배포된 Kubernetes 클러스터.
- GKE 클러스터에 구성된 워크로드 ID와 노드 풀에 구성된 GKE 메타데이터 서버.
- Google Cloud NetApp Volumes 관리자(roles/netapp.admin) 역할 또는 사용자 지정 역할이 있는 GCP 서비스 계정.
- "GCP"를 지정하는 cloudProvider와 새로운 GCP 서비스 계정을 지정하는 cloudIdentity를 포함하는 Trident

설치되었습니다. 아래에 예를 들어보겠습니다.

Trident 연산자

Trident 연산자를 사용하여 Trident 설치하려면 다음을 편집하세요. `tridentorchestrator_cr.yaml` 설정하다 `cloudProvider` 에게 "GCP" 그리고 설정하다 `cloudIdentity` 에게 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com` .

예를 들어:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

지배

다음 환경 변수를 사용하여 **cloud-provider (CP)** 및 **cloud-identity (CI)** 플래그의 값을 설정합니다.

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloudProvider` 환경 변수를 사용하여 GCP에 `$CP` 그리고 설정합니다 `cloudIdentity` 환경 변수 사용 `$ANNOTATION` :

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>트라이던트ctl</code>

다음 환경 변수를 사용하여 클라우드 공급자 및 클라우드 **ID** 플래그의 값을 설정합니다.

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

다음 예제에서는 Trident 설치하고 설정합니다. `cloud-provider` 플래그를 `$CP` , 그리고 `cloud-identity` 에게 `$ANNOTATION` :

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

Google Cloud NetApp Volumes 백엔드 구성을 준비하세요

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 요구 사항이 충족되는지 확인해야 합니다.

NFS 볼륨의 전제 조건

Google Cloud NetApp Volumes 처음 사용하거나 새로운 위치에서 사용하는 경우 Google Cloud NetApp Volumes 설정하고 NFS 볼륨을 생성하기 위해 초기 구성이 필요합니다. 참조하다"시작하기 전에" .

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 사항이 있는지 확인하세요.

- Google Cloud NetApp Volumes 서비스로 구성된 Google Cloud 계정입니다. 참조하다"Google Cloud NetApp Volumes" .
- Google Cloud 계정의 프로젝트 번호입니다. 참조하다"프로젝트 식별" .
- NetApp Volumes Admin이 있는 Google Cloud 서비스 계정(`roles/netapp.admin`) 역할. 참조하다"ID 및 액세스 관리 역할 및 권한" .
- GCNV 계정의 API 키 파일입니다. 참조하다"서비스 계정 키 생성"
- 저장 풀. 참조하다"스토리지 풀 개요" .

Google Cloud NetApp Volumes 에 대한 액세스를 설정하는 방법에 대한 자세한 내용은 다음을 참조하세요."Google Cloud NetApp Volumes 에 대한 액세스 설정" .

Google Cloud NetApp Volumes 백엔드 구성 옵션 및 예시

Google Cloud NetApp Volumes 의 백엔드 구성 옵션에 대해 알아보고 구성 예를 검토하세요.

백엔드 구성 옵션

각 백엔드는 단일 Google Cloud 지역에서 볼륨을 프로비저닝합니다. 다른 지역에 볼륨을 생성하려면 추가 백엔드를 정의할 수 있습니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	의 가치 `storageDriverName` "google-cloud-netapp-volumes"로 지정해야 합니다.
`backendName`	(선택 사항) 스토리지 백엔드의 사용자 정의 이름	드라이버 이름 + "_" + API 키의 일부

매개변수	설명	기본
`storagePools`	볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호. 해당 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	
`location`	Trident GCNV 볼륨을 생성하는 Google Cloud 위치입니다. 지역 간 Kubernetes 클러스터를 생성할 때 볼륨이 생성됩니다. `location` 여러 Google Cloud 지역의 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.	
`apiKey`	Google Cloud 서비스 계정에 대한 API 키 `netapp.admin` 역할. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다 (백엔드 구성 파일에 그대로 복사됨). 그만큼 `apiKey` 다음 키에 대한 키-값 쌍을 포함해야 합니다. `type` , `project_id` , `client_email` , `client_id` , `auth_uri` , `token_uri` , `auth_provider_x509_cert_url` , 그리고 `client_x509_cert_url` .	
`nfsMountOptions`	NFS 마운트 옵션에 대한 세부적인 제어.	"nfsvers=3"
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`serviceLevel`	스토리지 풀의 서비스 수준과 볼륨입니다. 값은 다음과 같습니다 `flex` , `standard` , `premium` , 또는 `extreme` .	
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`network`	GCNV 볼륨에 Google Cloud 네트워크가 사용됩니다.	
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, `{"api":false, "method":true}` . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 `nfs` , `smb` 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" . 예를 들어: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

볼륨 프로비저닝 옵션

기본 볼륨 프로비저닝을 제어할 수 있습니다. `defaults` 구성 파일의 섹션.

매개변수	설명	기본
`exportRule`	새로운 볼륨에 대한 내보내기 규칙. IPv4 주소의 조합을 쉼표로 구분하여 나열해야 합니다.	"0.0.0.0/0"
`snapshotDir`	에 대한 액세스 `.snapshot` 예배 규칙서	NFSv4의 경우 "true", NFSv3의 경우 "false"
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 백분율	"" (기본값 0 허용)
`unixPermissions`	새로운 볼륨의 유닉스 권한(8진수 4자리).	""

구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 백엔드의 최소 구성입니다. 이 구성을 사용하면 Trident 구성된 위치에서 Google Cloud NetApp Volumes에 위임된 모든 스토리지 풀을 검색하고 해당 풀 중 하나에 무작위로 새 볼륨을 배치합니다. 왜냐하면 `nasType` 생략됩니다. `nfs` 기본값이 적용되고 백엔드가 NFS 볼륨을 프로비저닝합니다.

이 구성은 Google Cloud NetApp Volumes 처음 시작하고 여러 가지를 시도해 볼 때 이상적이지만 실제로는 프로비저닝하는 볼륨에 대한 추가 범위를 제공해야 할 가능성이 높습니다.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

SMB 볼륨 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

StoragePools 필터를 사용한 구성

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 개의 가상 풀을 정의합니다. 가상 풀은 다음에 정의됩니다. `storage` 부분. 여러 개의 스토리지 풀이 서로 다른 서비스 수준을 지원하고 이를 나타내는 Kubernetes의 스토리지 클래스를 만들려는 경우에 유용합니다. 가상 풀 레이블은 풀을 구별하는 데 사용됩니다. 예를 들어, 아래 예에서 `performance` 라벨 및 `serviceLevel` 유형은 가상 풀을 구별하는 데 사용됩니다.

모든 가상 풀에 적용할 수 있는 기본값을 설정하고, 개별 가상 풀의 기본값을 덮어쓸 수도 있습니다. 다음 예에서, `snapshotReserve` 그리고 `exportRule` 모든 가상 풀의 기본값으로 사용됩니다.

자세한 내용은 다음을 참조하세요."가상 풀" .

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
```

```
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKE용 클라우드 ID

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

지원되는 토폴로지 구성

Trident 지역 및 가용성 영역에 따라 워크로드에 대한 볼륨 프로비저닝을 용이하게 합니다. 그만큼 `supportedTopologies` 이 백엔드 구성의 블록은 백엔드당 지역 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 지역 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 지역 및 영역 값과 일치해야 합니다. 이러한 지역과 영역은 저장 클래스에서 제공될 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공하는 지역 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우, Trident 언급된 지역 및 영역에 볼륨을 생성합니다. 자세한 내용은 다음을 참조하세요."CSI 토폴로지 사용" .

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

다음은 무엇인가요?

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
kubectl create -f <backend-file>
```

백엔드가 성공적으로 생성되었는지 확인하려면 다음 명령을 실행하세요.

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 백엔드를 사용하여 설명할 수 있습니다. `kubectl get tridentbackendconfig <backend-name>` 다음 명령을 실행하여 원인을 파악하려면 명령을 실행하거나 로그를 확인하세요.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후 백엔드를 삭제하고 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 정의

다음은 기본입니다 `StorageClass` 위의 백엔드를 참조하는 정의입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

- 다음을 사용한 정의 예 `parameter.selector` 필드:*

사용 중 `parameter.selector` 각각에 대해 지정할 수 있습니다 `StorageClass` 그만큼"가상 풀" 볼륨을 호스팅하는 데 사용됩니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

저장 클래스에 대한 자세한 내용은 다음을 참조하세요."스토리지 클래스 생성" .

SMB 볼륨에 대한 예제 정의

사용 중 `nasType` , `node-stage-secret-name` , 그리고 `node-stage-secret-namespace` SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다. 노드 단계 비밀에는 권한이 있거나 없는 모든 Active Directory 사용자/비밀번호를 사용할 수 있습니다.

기본 네임스페이스의 기본 구성

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

네임스페이스별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

볼륨별로 다른 비밀 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`SMB 볼륨을 지원하는 풀에 대한 필터입니다. `nasType: nfs 또는 nasType: null NFS 풀에 대한 필터.

PVC 정의 예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

PVC가 바인딩되었는지 확인하려면 다음 명령을 실행하세요.

```
kubectl get pvc gcnv-nfs-pvc

NAME           STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc   Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213   100Gi
RWX            gcnv-nfs-sc  1m
```

Google Cloud 백엔드에 대한 Cloud Volumes Service 구성

제공된 샘플 구성을 사용하여 Trident 설치의 백엔드로 Google Cloud용 NetApp Cloud Volumes Service 구성하는 방법을 알아보세요.

Google Cloud 드라이버 세부 정보

Trident 다음을 제공합니다. `gcp-cvs` 클러스터와 통신하는 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`gcp-cvs`	NFS	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`

Google Cloud용 Cloud Volumes Service 에 대한 Trident 지원에 대해 알아보세요.

Trident 두 가지 중 하나에서 Cloud Volumes Service 볼륨을 생성할 수 있습니다."서비스 유형" :

- **CVS-Performance**: 기본 Trident 서비스 유형입니다. 이러한 성능 최적화된 서비스 유형은 성능을 중시하는 프로덕션 워크로드에 가장 적합합니다. CVS-Performance 서비스 유형은 최소 100GiB 크기의 볼륨을 지원하는 하드웨어 옵션입니다. 다음 중 하나를 선택할 수 있습니다."세 가지 서비스 수준" :
  - `standard`
  - `premium`
  - `extreme`
- **CVS**: CVS 서비스 유형은 제한적에서 중간 수준의 성능 수준으로 높은 지역적 가용성을 제공합니다. CVS 서비스 유형은 스토리지 풀을 사용하여 최소 1GiB의 볼륨을 지원하는 소프트웨어 옵션입니다. 스토리지 풀은 최대 50개의 볼륨을 포함할 수 있으며, 모든 볼륨은 풀의 용량과 성능을 공유합니다. 다음 중 하나를 선택할 수 있습니다."두 가지 서비스 수준" :
  - `standardsw`
  - `zoneredundantstandardsw`

필요한 것

구성하고 사용하려면 "Google Cloud용 Cloud Volumes Service" 백엔드에는 다음이 필요합니다.

- NetApp Cloud Volumes Service 로 구성된 Google Cloud 계정
- Google Cloud 계정의 프로젝트 번호
- Google Cloud 서비스 계정 `netappcloudvolumes.admin` 역할
- Cloud Volumes Service 계정의 API 키 파일

백엔드 구성 옵션

각 백엔드는 단일 Google Cloud 지역에서 볼륨을 프로비저닝합니다. 다른 지역에 볼륨을 생성하려면 추가 백엔드를 정의할 수 있습니다.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	"gcp-cvs"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + API 키의 일부
`storageClass`	CVS 서비스 유형을 지정하는 데 사용되는 선택적 매개변수입니다. 사용 `software` CVS 서비스 유형을 선택하세요. 그렇지 않으면 Trident CVS-Performance 서비스 유형을 가정합니다.(`hardware`).	
`storagePools`	CVS 서비스 유형만 해당. 볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호. 해당 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	
`hostProjectNumber`	공유 VPC 네트워크를 사용하는 경우 필요합니다. 이 시나리오에서는 `projectNumber` 서비스 프로젝트이고, `hostProjectNumber` 호스트 프로젝트입니다.	

매개변수	설명	기본
apiRegion	Trident Cloud Volumes Service 볼륨을 생성하는 Google Cloud 지역입니다. 지역 간 Kubernetes 클러스터를 생성할 때 볼륨이 생성됩니다. apiRegion 여러 Google Cloud 지역의 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.	
apiKey	Google Cloud 서비스 계정에 대한 API 키 netappcloudvolumes.admin 역할. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다(백엔드 구성 파일에 그대로 복사됨).	
proxyURL	CVS 계정에 연결하는 데 프록시 서버가 필요한 경우 프록시 URL입니다. 프록시 서버는 HTTP 프록시 또는 HTTPS 프록시가 될 수 있습니다. HTTPS 프록시의 경우 인증서 유효성 검사를 건너뛰어 프록시 서버에서 자체 서명된 인증서를 사용할 수 있습니다. 인증이 활성화된 프록시 서버는 지원되지 않습니다.	
nfsMountOptions	NFS 마운트 옵션에 대한 세부적인 제어.	"nfsvers=3"
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
serviceLevel	새로운 볼륨에 대한 CVS-Performance 또는 CVS 서비스 수준입니다. CVS-Performance 값은 다음과 같습니다. standard , premium , 또는 extreme . CVS 값은 standardsw 또는 zoneredundantstandardsw .	CVS-Performance의 기본값은 "표준"입니다. CVS 기본값은 "standardsw"입니다.
network	Cloud Volumes Service 볼륨에 사용되는 Google Cloud 네트워크입니다.	"기본"
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예, \{"api":false, "method":true} . 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면 이 기능을 사용하지 마세요.	널
allowedTopologies	지역 간 액세스를 활성화하려면 StorageClass 정의를 다음과 같이 합니다. allowedTopologies 모든 지역을 포함해야 합니다. 예를 들어: - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

볼륨 프로비저닝 옵션

기본 볼륨 프로비저닝을 제어할 수 있습니다. defaults 구성 파일의 섹션.

매개변수	설명	기본
exportRule	새로운 볼륨에 대한 내보내기 규칙. CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷의 조합을 쉼표로 구분한 목록이어야 합니다.	"0.0.0.0/0"

매개변수	설명	기본
`snapshotDir`	에 대한 액세스 `.snapshot` 예배 규칙서	"거짓"
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 백분율	"" (CVS 기본값 0 허용)
`size`	새로운 볼륨의 크기. CVS-Performance 최소값은 100GiB입니다. CVS 최소 크기는 1GiB입니다.	CVS-Performance 서비스 유형은 기본적으로 "100GiB"로 설정됩니다. CVS 서비스 유형은 기본값을 설정하지 않지만 최소 1GiB가 필요합니다.

CVS-Performance 서비스 유형 예

다음 예제에서는 CVS-Performance 서비스 유형에 대한 샘플 구성을 제공합니다.

예 **1:** 최소 구성

이는 기본 "표준" 서비스 수준을 갖춘 기본 CVS-Performance 서비스 유형을 사용하는 최소 백엔드 구성입니다.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: "012345678901"
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: <id_value>
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: "123456789012345678901"
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

예 **2:** 서비스 수준 구성

이 샘플은 서비스 수준, 볼륨 기본값을 포함한 백엔드 구성 옵션을 보여줍니다.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

예제 **3:** 가상 풀 구성

이 샘플은 다음을 사용합니다. `storage` 가상 풀을 구성하려면 `StorageClasses` 그것들을 다시 언급하는 것. 참조하다스토리지 클래스 정의 저장 클래스가 어떻게 정의되었는지 확인하세요.

여기서 모든 가상 풀에 대한 특정 기본값이 설정됩니다. `snapshotReserve` 5%에서 그리고 `exportRule` 0.0.0.0/0으로. 가상 풀은 다음에 정의됩니다. `storage` 부분. 각 개별 가상 풀은 자체적으로 정의합니다. `serviceLevel` 일부 풀은 기본값을 덮어씁니다. 가상 풀 레이블은 다음을 기준으로 풀을 구별하는 데 사용되었습니다. `performance` 그리고 `protection` .

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
```

```
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
  serviceLevel: standard
```

스토리지 클래스 정의

다음 StorageClass 정의는 가상 풀 구성 예제에 적용됩니다. 사용 중 `parameters.selector` 각 StorageClass에 대해 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨에는 선택된 풀에서 정의된 측면이 있습니다.

스토리지 클래스 예제

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=extra
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium; protection=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
```

```
  selector: performance=standard
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: protection=extra
allowVolumeExpansion: true
```

- 첫 번째 StorageClass(`cvs-extreme-extra-protection`)는 첫 번째 가상 풀에 매핑됩니다. 이 풀은 스냅샷 리저브가 10%로 극한의 성능을 제공하는 유일한 풀입니다.
- 마지막 StorageClass(`cvs-extra-protection`)는 10%의 스냅샷 예약을 제공하는 모든 스토리지 풀을 호출합니다. Trident 어떤 가상 풀을 선택할지 결정하고 스냅샷 예약 요구 사항이 충족되는지 확인합니다.

CVS 서비스 유형 예시

다음 예제에서는 CVS 서비스 유형에 대한 샘플 구성을 제공합니다.

예 **1:** 최소 구성

이는 다음을 사용하는 최소 백엔드 구성입니다. `storageClass` CVS 서비스 유형 및 기본값을 지정하려면 `standardsw` 서비스 수준.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

예 **2:** 스토리지 풀 구성

이 샘플 백엔드 구성은 다음을 사용합니다. `storagePools` 스토리지 풀을 구성하려면.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

다음은 무엇인가요**?**

백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

NetApp HCI 또는 SolidFire 백엔드 구성

Trident 설치로 Element 백엔드를 만들고 사용하는 방법을 알아보세요.

요소 드라이버 세부 정보

Trident 다음을 제공합니다. `solidfire-san` 클러스터와 통신하기 위한 저장 드라이버. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

그만큼 `solidfire-san` 저장 드라이버는 파일 및 블록 볼륨 모드를 지원합니다. 를 위해 `Filesystem` volumeMode, Trident 볼륨을 생성하고 파일 시스템을 생성합니다. 파일 시스템 유형은 StorageClass에 의해 지정됩니다.

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`solidfire-san`	iSCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템이 없습니다. 원시 블록 장치.
`solidfire-san`	iSCSI	파일 시스템	RWO, RWOP	`xfs`, `ext3` , `ext4`

시작하기 전에

Element 백엔드를 생성하려면 다음이 필요합니다.

- Element 소프트웨어를 실행하는 지원되는 저장 시스템입니다.
- 볼륨을 관리할 수 있는 NetApp HCI/ SolidFire 클러스터 관리자 또는 테넌트 사용자에 대한 자격 증명입니다.
- 모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 참조하다"워커 노드 준비 정보" .

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	항상 "solidfire-san"
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	"solidfire_" + 스토리지(iSCSI) IP 주소
`Endpoint`	테넌트 자격 증명을 사용한 SolidFire 클러스터용 MVIP	

매개변수	설명	기본
`SVIP`	스토리지(iSCSI) IP 주소 및 포트	
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 집합입니다.	""
`TenantName`	사용할 테넌트 이름(찾을 수 없는 경우 생성)	
`InitiatorIFace`	iSCSI 트래픽을 특정 호스트 인터페이스로 제한	"기본"
`UseCHAP`	CHAP를 사용하여 iSCSI를 인증합니다. Trident CHAP를 사용합니다.	true
`AccessGroups`	사용할 액세스 그룹 ID 목록	"trident"라는 액세스 그룹의 ID를 찾습니다.
`Types`	QoS 사양	
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}	널

사용하지 마십시오 `debugTraceFlags` 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.

예 **1:** 백엔드 구성 `solidfire-san` **3**가지 볼륨 유형을 갖춘 드라이버

이 예제에서는 CHAP 인증을 사용하고 특정 QoS 보장을 갖춘 세 가지 볼륨 유형을 모델링하는 백엔드 파일을 보여줍니다. 그러면 각각을 사용하기 위해 저장소 클래스를 정의할 가능성이 가장 높습니다. `IOPS` 저장 클래스 매개변수.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

예 **2:** 백엔드 및 스토리지 클래스 구성 `solidfire-san` 가상 풀이 있는 드라이버

이 예에서는 가상 풀과 이를 참조하는 StorageClass로 구성된 백엔드 정의 파일을 보여줍니다.

Trident 프로비저닝 시 스토리지 풀에 있는 레이블을 백엔드 스토리지 LUN에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값이 설정되어 있습니다. `type` 실버에서. 가상 풀은 다음에 정의됩니다. `storage` 부분. 이 예에서 일부 스토리지 풀은 자체 유형을 설정하고, 일부 풀은 위에 설정된 기본값을 재정의합니다.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
```

```
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

다음 StorageClass 정의는 위의 가상 풀을 참조합니다. 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

첫 번째 StorageClass(`solidfire-gold-four`)은 첫 번째 가상 풀에 매핑됩니다. 이것은 골드 성능을 제공하는 유일한 풀입니다. `Volume Type QoS` 금의. 마지막 StorageClass(`solidfire-silver`)은 실버 성능을 제공하는 모든 스토리지 풀을 호출합니다. Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

더 많은 정보를 찾아보세요

- "볼륨 액세스 그룹"

ONTAP SAN 드라이버

ONTAP SAN 드라이버 개요

ONTAP 및 Cloud Volumes ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보세요.

ONTAP SAN 드라이버 세부 정보

Trident ONTAP 클러스터와 통신하기 위해 다음과 같은 SAN 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	FC를 통한 iSCSI SCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san`	FC를 통한 iSCSI SCSI	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`
`ontap-san`	NVMe/TCP 참조하다NVMe/TCP에 대한 추가 고려 사항 .	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san`	NVMe/TCP 참조하다NVMe/TCP에 대한 추가 고려 사항 .	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san-economy`	iSCSI	차단하다	RWO, ROX, RWX, RWOP	파일 시스템 없음; 원시 블록 장치
`ontap-san-economy`	iSCSI	파일 시스템	RWO, RWOP ROX와 RWX는 파일 시스템 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3` , `ext4`

- 사용 `ontap-san-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" .
- 사용 `ontap-nas-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 그리고 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마세요 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 모빌리티가 필요할 것으로 예상되는 경우
- NetApp ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 증가를 사용하지 않는 것을 권장합니다. 이 문제를 해결하기 위해 Trident 스냅샷 예약 사용을 지원하고 Flexvol 볼륨을 그에 따라 확장합니다.

사용자 권한

Trident 일반적으로 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. Amazon FSx for NetApp ONTAP 배포의 경우 Trident 클러스터를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. 그만큼 `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

당신이 사용하는 경우 `limitAggregateUsage` 매개변수, 클러스터 관리자 권한이 필요합니다. Trident 와 함께 Amazon FSx for NetApp ONTAP 사용하는 경우 `limitAggregateUsage` 매개변수는 작동하지 않습니다 `vsadmin` 그리고 `fsxadmin` 사용자 계정. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만드는 것이 가능하지만 권장하지는 않습니다. Trident 의 새로운 릴리스 대부분은 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

NVMe/TCP에 대한 추가 고려 사항

Trident 다음을 사용하여 NVMe(Non-Volatile Memory Express) 프로토콜을 지원합니다. `ontap-san` 운전자 포함:

- IPv6
- NVMe 볼륨의 스냅샷 및 복제본
- NVMe 볼륨 크기 조정
- Trident 외부에서 생성된 NVMe 볼륨을 가져와서 Trident 에서 수명 주기를 관리할 수 있도록 합니다.
- NVMe 네이티브 멀티패싱

- K8s 노드의 정상적 또는 비정상적 종료(24.06)

Trident 다음을 지원하지 않습니다.

- NVMe에서 기본적으로 지원되는 DH-HMAC-CHAP
- 장치 매퍼(DM) 다중 경로
- LUKS 암호화

NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다.

ONTAP SAN 드라이버로 백엔드 구성을 준비합니다**.**

ONTAP SAN 드라이버로 ONTAP 백엔드를 구성하기 위한 요구 사항과 인증 옵션을 이해합니다.

요구 사항

모든 ONTAP 백엔드의 경우 Trident 최소한 하나의 집계가 SVM에 할당되어야 합니다.

"ASA r2 시스템"다른 ONTAP 시스템(ASA, AFF 및 FAS)과 저장 계층 구현 방식이 다릅니다. ASA r2 시스템에서는 집계 대신 스토리지 가용성 영역이 사용됩니다. 참조하다"이것" ASA r2 시스템에서 SVM에 집계를 할당하는 방법에 대한 지식 기반 문서입니다.

두 개 이상의 드라이버를 실행하고, 하나 또는 다른 하나를 가리키는 스토리지 클래스를 생성할 수도 있습니다. 예를 들어 다음을 구성할 수 있습니다. `san-dev` 를 사용하는 클래스 `ontap-san` 운전자와 `san-default` 를 사용하는 클래스 `ontap-san-economy` 하나.

모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 참조하다 "워커 노드 준비" 자세한 내용은.

ONTAP 백엔드 인증

Trident ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름과 비밀번호입니다. 다음과 같은 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다. `admin` 또는 `vsadmin` ONTAP 버전과의 최대 호환성을 보장합니다.
- 인증서 기반: Trident 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 여기서 백엔드 정의에는 클라이언트 인증서, 키, 신뢰할 수 있는 CA 인증서(사용되는 경우)의 Base64 인코딩 값이 포함되어야 합니다(권장).

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 사이를 전환할 수 있습니다. 하지만 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

*자격 증명과 인증서*를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. 다음과 같은 표준 사전 정의된 역할을 활용하는 것이 좋습니다. `admin` 또는 `vsadmin` . 이를 통해 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 공개할 수 있는 향후 ONTAP 릴리스와의 호환성이 보장됩니다. Trident 사용하면 사용자 정의 보안 로그인 역할을 만들어 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 곳이라는 점을 명심하세요. 백엔드가 생성된 후 사용자 이름/비밀번호는 Base64로 인코딩되어 Kubernetes 비밀로 저장됩니다. 백엔드를 만들거나 업데이트하는 것은 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행해야 하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

새로운 백엔드와 기존 백엔드는 인증서를 사용하고 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 개의 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64로 인코딩된 값입니다.
- clientPrivateKey: 연관된 개인 키의 Base64 인코딩된 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않으면 무시할 수 있습니다.

일반적인 작업 흐름은 다음 단계로 구성됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성할 때, 인증할 ONTAP 사용자로 일반 이름(CN)을 설정합니다.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 이 문제는 이미 스토리지 관리자가 처리했을 수도 있습니다. 신뢰할 수 있는 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서와 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 지원되는지 확인하세요. `cert` 인증방법.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. < ONTAP 관리 LIF> 및 <vserver 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법 업데이트 또는 자격 증명 순환

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 순환할 수 있습니다. 이는 양방향으로 작동합니다. 사용자 이름/비밀번호를 사용하는 백엔드는 인증서를 사용하도록 업데이트할 수 있고, 인증서를 활용하는 백엔드는 사용자 이름/비밀번호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새로운 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수를 포함하는 업데이트된 backend.json 파일을 사용하여 실행합니다. `tridentctl backend update` .

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

비밀번호를 순환할 때 스토리지 관리자는 먼저 ONTAP 에서 사용자의 비밀번호를 업데이트해야 합니다. 이어서 백엔드 업데이트가 진행됩니다. 인증서를 순환할 때 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 그런 다음 백엔드를 업데이트하여 새 인증서를 사용하고, 그 후 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 만들어진 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공하면 Trident ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident 에 대한 사용자 정의 ONTAP 역할 생성

Trident 에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용하지 않아도 되도록 최소한의 권한으로 ONTAP 클러스터 역할을 만들 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident 사용자가 만든 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

참조하다"Trident 사용자 정의 역할 생성기" Trident 사용자 정의 역할 생성에 대한 자세한 내용은 다음을 참조하세요.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 만듭니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자에 대한 사용자 이름을 만듭니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할을 매핑합니다.

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

시스템 관리자 사용

ONTAP 시스템 관리자에서 다음 단계를 수행합니다.

1. 사용자 정의 역할 만들기:
   a. 클러스터 수준에서 사용자 지정 역할을 만들려면 *클러스터 > 설정*을 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 만들려면 저장소 **>** 저장소 **VM > `required SVM` >** 설정 **>** 사용자 및 역할.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. *역할*에서 *+추가*를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. * Trident 사용자에게 역할 매핑*: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에 있는 추가 아이콘 *+*을 선택합니다.
   b. 필요한 사용자 이름을 선택하고, 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭하세요.

자세한 내용은 다음 페이지를 참조하세요.

- "ONTAP 관리를 위한 사용자 정의 역할"또는"사용자 정의 역할 정의"
- "역할 및 사용자 작업"

양방향 **CHAP**를 사용하여 연결 인증

Trident 양방향 CHAP를 사용하여 iSCSI 세션을 인증할 수 있습니다. `ontap-san` 그리고 `ontap-san-economy` 운전자. 이를 위해서는 다음을 활성화해야 합니다. `useCHAP` 백엔드 정의에 옵션을 추가하세요. 설정 시 `true` Trident SVM의 기본 이니시에이터 보안을 양방향 CHAP로 구성하고 백엔드 파일에서 사용자 이름과 비밀번호를 설정합니다.

NetApp 양방향 CHAP를 사용하여 연결을 인증할 것을 권장합니다. 다음 샘플 구성을 참조하세요.

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

그만큼 `useCHAP` 매개변수는 한 번만 구성할 수 있는 부울 옵션입니다. 기본적으로 false로 설정됩니다. true로 설정한 후에는 false로 설정할 수 없습니다.

또한 `useCHAP=true` , 그 `chapInitiatorSecret` , `chapTargetInitiatorSecret` , `chapTargetUsername` , 그리고 `chapUsername` 필드는 백엔드 정의에 포함되어야 합니다. 백엔드가 생성된 후 다음을 실행하여 비밀을 변경할 수 있습니다. `tridentctl update` .

작동 원리

설정하여 `useCHAP` true로 설정하면 스토리지 관리자가 Trident 스토리지 백엔드에서 CHAP를 구성하도록 지시합니다. 여기에는 다음이 포함됩니다.

- SVM에 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 없음(기본적으로 설정됨)이고 볼륨에 이미 존재하는 LUN이 없는 경우 Trident 기본 보안 유형을 다음과 같이 설정합니다. `CHAP` CHAP 이니시에이터와 대상 사용자 이름 및 비밀번호를 구성합니다.
  - SVM에 LUN이 포함되어 있으면 Trident SVM에서 CHAP를 활성화하지 않습니다. 이렇게 하면 SVM에 이미 있는 LUN에 대한 액세스가 제한되지 않습니다.
- CHAP 이니시에이터와 대상 사용자 이름 및 비밀번호를 구성합니다. 이러한 옵션은 백엔드 구성에서 지정해야 합니다(위에 표시된 대로).

백엔드가 생성된 후 Trident 해당 백엔드를 생성합니다. `tridentbackend` CRD를 사용하여 CHAP 비밀과 사용자 이름을 Kubernetes 비밀로 저장합니다. 이 백엔드에서 Trident 가 생성한 모든 PV는 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 순환하고 백엔드를 업데이트합니다.

CHAP 매개변수를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있습니다. `backend.json` 파일. 이렇게 하려면 CHAP 비밀을 업데이트하고 다음을 사용해야 합니다. `tridentctl update` 이러한 변경 사항을 반영하기 위한 명령입니다.

백엔드의 CHAP 비밀을 업데이트할 때는 다음을 사용해야 합니다. `tridentctl` 백엔드를 업데이트합니다. Trident 이러한 변경 사항을 선택할 수 없으므로 ONTAP CLI 또는 ONTAP 시스템 관리자를 사용하여 스토리지 클러스터의 자격 증명을 업데이트하지 마십시오.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

기존 연결은 영향을 받지 않으며, SVM에서 Trident 가 자격 증명을 업데이트하면 계속 활성 상태를 유지합니다. 새로운 연결은 업데이트된 자격 증명을 사용하고 기존 연결은 계속 활성 상태를 유지합니다. 이전 PV의 연결을 끊었다가 다시 연결하면 업데이트된 자격 증명을 사용하게 됩니다.

ONTAP SAN 구성 옵션 및 예

Trident 설치로 ONTAP SAN 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClass에 매핑하기 위한 세부 정보를 제공합니다.

"ASA r2 시스템"다른 ONTAP 시스템(ASA, AFF, FAS)과 저장 계층 구현 방식이 다릅니다. 이러한 변화는 표기된 특정 매개변수의 사용에 영향을 미칩니다. "ASA r2 시스템과 다른 ONTAP 시스템 간의 차이점에 대해 자세히 알아보세요.".

오직 `ontap-san` 드라이버(iSCSI 및 NVMe/TCP 프로토콜 포함)는 ASA r2 시스템에서 지원됩니다.

Trident 백엔드 구성에서는 시스템이 ASA r2라고 지정할 필요가 없습니다. 선택할 때 `ontap-san` 로서 `storageDriverName` Trident ASA r2 또는 기존 ONTAP 시스템을 자동으로 감지합니다. 아래 표에 나와 있는 것처럼 일부 백엔드 구성 매개변수는 ASA r2 시스템에 적용할 수 없습니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-san`또는 `ontap-san-economy`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 원활한 MetroCluster 전환을 위해서는 다음을 참조하세요 .MetroCluster 예제 . "vsadmin" 자격 증명을 사용하는 경우 `managementLIF` SVM의 자격 증명이어야 합니다. "관리자" 자격 증명을 사용하는 경우 `managementLIF` 클러스터의 것이어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	프로토콜 LIF의 IP 주소. Trident IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . **iSCSI**에 대해서는 지정하지 마세요**.** Trident 사용"ONTAP 선택적 LUN 맵" 다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. 경고가 생성됩니다. `dataLIF` 명확하게 정의되어 있습니다. 메트로클러스터는 생략합니다**.** 를 참조하십시오MetroCluster 예제 .	SVM에 의해 파생됨
`svm`	사용할 스토리지 가상 머신 **Metrocluster**의 경우 생략 를 참조하십시오MetroCluster 예제 .	SVM의 경우 파생됨 `managementLIF` 지정됨

매개변수	설명	기본
`useCHAP`	CHAP를 사용하여 ONTAP SAN 드라이버에 대한 iSCSI를 인증합니다[부울]. 로 설정 `true` Trident 백엔드에 제공된 SVM에 대한 기본 인증으로 양방향 CHAP를 구성하고 사용하도록 합니다. 참조하다 "ONTAP SAN 드라이버로 백엔드 구성을 준비합니다." 자세한 내용은. **FCP** 또는 **NVMe/TCP**에서는 지원되지 않습니다.	`false`
`chapInitiatorSecret`	CHAP 개시자 비밀. 필요한 경우 `useCHAP=true`	""
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`chapTargetInitiatorSecret`	CHAP 대상 개시자 비밀. 필요한 경우 `useCHAP=true`	""
`chapUsername`	수신 사용자 이름. 필요한 경우 `useCHAP=true`	""
`chapTargetUsername`	대상 사용자 이름. 필요한 경우 `useCHAP=true`	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됩니다.	""
`username`	ONTAP 클러스터와 통신하려면 사용자 이름이 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
`password`	ONTAP 클러스터와 통신하려면 비밀번호가 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
`svm`	사용할 스토리지 가상 머신	SVM의 경우 파생됨 `managementLIF` 지정됨
`storagePrefix`	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 나중에 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새로운 백엔드를 만들어야 합니다.	`trident`

매개변수	설명	기본
`aggregate`	프로비저닝을 위한 집계(선택 사항, 설정된 경우 SVM에 할당해야 함). 를 위해 `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당되지 않은 경우 사용 가능한 모든 집계를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 집계가 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계를 구성한 경우, 집계의 이름이 변경되거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 SVM에 있는 집계로 변경하거나 집계를 완전히 제거해야 합니다. • ASA r2 시스템에 대해서는 지정하지 마세요*.	""
`limitAggregateUsage`	사용량이 이 백분율을 초과하면 프로비저닝에 실패합니다. Amazon FSx for NetApp ONTAP 백엔드를 사용하는 경우 지정하지 마십시오. `limitAggregateUsage` . 제공된 `fsxadmin` 그리고 `vsadmin` Trident 사용하여 집계 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다. * ASA r2 시스템에 대해서는 지정하지 마세요*.	"" (기본적으로 적용되지 않음)
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	Flexvol당 최대 LUN은 [50, 200] 범위 내에 있어야 합니다.	`100`
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}는 문제를 해결하고 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마세요.	`null`

매개변수	설명	기본
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. `useREST`설정 시 `true` , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. `false` Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. `ontapi` 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. `vsadmin` 그리고 `cluster-admin` 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST` 로 설정됩니다 `true` 기본적으로; 변경 `useREST` 에게 `false` ONTAPI(ZAPI) 호출을 사용합니다. `useREST`NVMe/TCP에 완벽하게 적합합니다. NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다. *지정된 경우 항상 다음으로 설정됩니다. true ASA r2 시스템*용.	true`ONTAP 9.15.1 이상인 경우, 그렇지 않은 경우 `false.
sanType	선택에 사용 iscsi iSCSI의 경우, nvme NVMe/TCP 또는 fcp FC(Fibre Channel)를 통한 SCSI의 경우.	`iscsi`비어있는 경우
formatOptions	사용 formatOptions 명령줄 인수를 지정하려면 mkfs 볼륨이 포맷될 때마다 적용되는 명령입니다. 이를 통해 사용자의 선호도에 따라 볼륨을 포맷할 수 있습니다. 장치 경로를 제외하고 mkfs 명령 옵션과 비슷하게 formatOptions를 지정해야 합니다. 예: "-E nodiscard" 지원됨 **ontap-san** 그리고 **ontap-san-economy** **iSCSI** 프로토콜을 사용하는 드라이버. 또한 **iSCSI** 및 **NVMe/TCP** 프로토콜을 사용하는 **ASA r2** 시스템에서도 지원됩니다.	
limitVolumePoolSize	ontap-san-economy 백엔드에서 LUN을 사용할 때 요청 가능한 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본
`denyNewVolumePools`	제한하다 `ontap-san-economy` 백엔드가 LUN을 포함하기 위해 새로운 FlexVol 볼륨을 생성하지 못하도록 합니다. 새로운 PV를 프로비저닝하는 데는 기존 Flexvol만 사용됩니다.	

formatOptions 사용에 대한 권장 사항

Trident 포맷 과정을 신속하게 진행하기 위해 다음 옵션을 권장합니다.

-E 삭제 안 함**:**

- 유지, mkfs 시점에 블록을 삭제하려고 시도하지 마세요(처음에 블록을 삭제하는 것은 솔리드 스테이트 장치와 스파스/씬 프로비저닝 스토리지에서 유용합니다). 이는 더 이상 사용되지 않는 옵션인 "-K"를 대체하며 모든 파일 시스템(xfs, ext3, ext4)에 적용할 수 있습니다.

Active Directory 자격 증명을 사용하여 백엔드 **SVM**에 **Trident** 인증

Active Directory(AD) 자격 증명을 사용하여 백엔드 SVM에 인증하도록 Trident 구성할 수 있습니다. AD 계정이 SVM에 액세스하려면 먼저 클러스터 또는 SVM에 대한 AD 도메인 컨트롤러 액세스를 구성해야 합니다. AD 계정으로 클러스터를 관리하려면 도메인 터널을 만들어야 합니다. 참조하다 "ONTAP 에서 Active Directory 도메인 컨트롤러 액세스 구성" 자세한 내용은.

단계

1. 백엔드 SVM에 대한 DNS(도메인 이름 시스템) 설정을 구성합니다.

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Active Directory에서 SVM에 대한 컴퓨터 계정을 만들려면 다음 명령을 실행하세요.

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. 이 명령을 사용하여 클러스터 또는 SVM을 관리할 AD 사용자 또는 그룹을 만듭니다.

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Trident 백엔드 구성 파일에서 다음을 설정합니다. `username` 그리고 `password` 각각 AD 사용자 또는 그룹 이름과 비밀번호에 대한 매개변수입니다.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. `defaults` 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
`spaceAllocation`	LUN에 대한 공간 할당	"true" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
`spaceReserve`	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움). 설정 **none ASA r2** 시스템용.	"없음"
`snapshotPolicy`	사용할 스냅샷 정책입니다. *설정 `none` ASA r2 시스템*용.	"없음"
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용해야 하며, 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하세요.	""
`snapshotReserve`	스냅샷을 위해 예약된 볼륨의 비율입니다. * ASA r2 시스템에 대해서는 지정하지 마세요*.	"0"이면 `snapshotPolicy` "없음"이고, 그렇지 않으면 ""
`splitOnClone`	생성 시 부모로부터 복제본을 분할합니다.	"거짓"
`encryption`	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. `false` . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	"false" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
`luksEncryption`	LUKS 암호화를 활성화합니다. 참조하다"Linux Unified Key Setup(LUKS) 사용" .	"" *로 설정 `false` ASA r2 시스템*용.
`tieringPolicy`	"없음"을 사용하는 계층화 정책 * ASA r2 시스템에는 지정하지 마세요*.	
`nameTemplate`	사용자 정의 볼륨 이름을 만드는 템플릿입니다.	""

볼륨 프로비저닝 예시

기본값이 정의된 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

다음을 사용하여 생성된 모든 볼륨의 경우 `ontap-san` 드라이버, Trident LUN 메타데이터를 수용하기 위해 FlexVol 에 10%의 추가 용량을 추가합니다. LUN은 PVC에서 사용자가 요청한 정확한 크기로 프로비저닝됩니다. Trident FlexVol 에 10%를 추가합니다(ONTAP 에서는 사용 가능한 크기로 표시됨). 이제 사용자는 요청한 사용 가능한 용량을 얻게 됩니다. 이 변경 사항은 사용 가능한 공간이 완전히 활용되지 않는 한 LUN이 읽기 전용이 되는 것을 방지합니다. 이는 ontap-san-economy에는 적용되지 않습니다.

정의하는 백엔드의 경우 `snapshotReserve` Trident 다음과 같이 볼륨의 크기를 계산합니다.

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Trident FlexVol 에 추가하는 10%입니다. 을 위한 `snapshotReserve` = 5%, PVC 요청 = 5GiB인 경우 총 볼륨 크기는 5.79GiB이고 사용 가능한 크기는 5.5GiB입니다. 그만큼 `volume show` 명령을 실행하면 다음 예와 비슷한 결과가 표시됩니다.

현재 기존 볼륨에 대한 새로운 계산을 사용하려면 크기 조정만이 유일한 방법입니다.

최소 구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

> Trident 와 함께 NetApp ONTAP 에서 Amazon FSx 사용하는 경우 NetApp IP 주소 대신 LIF에 DNS 이름을 지정할 것을 권장합니다.

ONTAP SAN 예시

이는 다음을 사용하는 기본 구성입니다. `ontap-san` 운전사.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

MetroCluster 예제

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다."SVM 복제 및 복구" .

원활한 전환 및 전환을 위해 다음을 사용하여 SVM을 지정하십시오. `managementLIF` 그리고 생략하다 `svm` 매개변수. 예를 들어:

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

ONTAP SAN 경제 사례

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

인증서 기반 인증 예제

이 기본 구성 예에서는 `clientCertificate` , `clientPrivateKey` , 그리고 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)이 채워집니다. `backend.json` 그리고 클라이언트 인증서, 개인 키, 신뢰할 수 있는 CA 인증서의 base64 인코딩 값을 각각 가져옵니다.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

양방향 **CHAP** 예시

이러한 예제는 백엔드를 생성합니다. `useCHAP` 로 설정 `true` .

ONTAP SAN CHAP 예시

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN 경제 **CHAP** 예시

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCP 예제

ONTAP 백엔드에 NVMe로 구성된 SVM이 있어야 합니다. 이는 NVMe/TCP의 기본 백엔드 구성입니다.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

FC(FCP)를 통한 **SCSI** 예제

ONTAP 백엔드에 FC가 구성된 SVM이 있어야 합니다. 이는 FC의 기본 백엔드 구성입니다.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

ontap-san-economy 드라이버에 대한 **formatOptions** 예제

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

가상 풀을 사용한 백엔드의 예

이러한 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 다음과 같은 특정 기본값이 설정됩니다. `spaceReserve` 없음, `spaceAllocation` 거짓이고, `encryption` 거짓. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident "주석" 필드에 프로비저닝 라벨을 설정합니다. FlexVol volume 에 대한 주석이 설정되면 Trident 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이러한 예에서 일부 스토리지 풀은 자체적으로 설정합니다. `spaceReserve` , `spaceAllocation` , 그리고 `encryption` 값이 지정되고 일부 풀은 기본값을 재정의합니다.

ONTAP SAN 예시

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

ONTAP SAN 경제 사례

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
```

```
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

NVMe/TCP 예제

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

백엔드를 **StorageClass**에 매핑

다음 StorageClass 정의는 다음을 참조합니다.가상 풀을 사용한 백엔드의 예 . 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

- 그만큼 `protection-gold` StorageClass는 첫 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이곳은 골드 레벨의 보호를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 그만큼 `protection-not-gold` StorageClass는 두 번째 및 세 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이것들은 금 이외의 보호 수준을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 그만큼 `app-mysqldb` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-san-economy` 백엔드. 이는 mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 그만큼 `protection-silver-creditpoints-20k` StorageClass는 두 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드. 이 풀은 실버 레벨 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 그만큼 `creditpoints-5k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-san` 백엔드와 네 번째 가상 풀 `ontap-san-economy` 백엔드. 5000 크레딧 포인트를 제공하는 유일한 풀 서비스입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- 그만큼 `my-test-app-sc` StorageClass는 다음에 매핑됩니다. `testAPP` 가상 풀 `ontap-san` 운전자와 함께 `sanType: nvme` . 이것은 유일한 풀 제공입니다 `testApp` .

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

ONTAP NAS 드라이버

ONTAP NAS 드라이버 개요

ONTAP 및 Cloud Volumes ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하는 방법에 대해 알아보세요.

ONTAP NAS 드라이버 세부 정보

Trident ONTAP 클러스터와 통신하기 위해 다음과 같은 NAS 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 다음과 같습니다: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

운전사	규약	볼륨모드	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-nas`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`
`ontap-nas-economy`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`
`ontap-nas-flexgroup`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	"", `nfs` , `smb`

- 사용 `ontap-san-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" .
- 사용 `ontap-nas-economy` 지속적 볼륨 사용 횟수가 다음보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 그리고 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마세요 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 모빌리티가 필요할 것으로 예상되는 경우
- NetApp ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 증가를 사용하지 않는 것을 권장합니다. 이 문제를 해결하기 위해 Trident 스냅샷 예약 사용을 지원하고 Flexvol 볼륨을 그에 따라 확장합니다.

사용자 권한

Trident 일반적으로 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다.

Amazon FSx for NetApp ONTAP 배포의 경우 Trident 클러스터를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상합니다. `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자입니다. 그만큼 `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

당신이 사용하는 경우 `limitAggregateUsage` 매개변수, 클러스터 관리자 권한이 필요합니다. Trident 와 함께 Amazon FSx for NetApp ONTAP 사용하는 경우 `limitAggregateUsage` 매개변수는 작동하지 않습니다 `vsadmin` 그리고 `fsxadmin` 사용자 계정. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 보다 제한적인 역할을 만드는 것이 가능하지만 권장하지는 않습니다. Trident 의 새로운 릴리스 대부분은 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

ONTAP NAS 드라이버로 백엔드 구성을 준비합니다.

ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항, 인증 옵션 및 내보내기 정책을 이해합니다.

요구 사항

- 모든 ONTAP 백엔드의 경우 Trident 최소한 하나의 집계가 SVM에 할당되어야 합니다.
- 두 개 이상의 드라이버를 실행하고, 하나를 가리키는 스토리지 클래스를 만들 수 있습니다. 예를 들어, 다음을 사용하는 Gold 클래스를 구성할 수 있습니다. `ontap-nas` 드라이버 및 Bronze 클래스를 사용하는 `ontap-nas-economy` 하나.
- 모든 Kubernetes 워커 노드에는 적절한 NFS 도구가 설치되어 있어야 합니다. 참조하다"여기" 자세한 내용은.
- Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 참조하다 SMB 볼륨 프로비저닝 준비 자세한 내용은.

ONTAP 백엔드 인증

Trident ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 이 모드에서는 ONTAP 백엔드에 대한 충분한 권한이 필요합니다. 다음과 같은 사전 정의된 보안 로그인 역할과 연결된 계정을 사용하는 것이 좋습니다. `admin` 또는 `vsadmin` ONTAP 버전과의 최대 호환성을 보장합니다.
- 인증서 기반: 이 모드에서는 Trident ONTAP 클러스터와 통신하려면 백엔드에 인증서가 설치되어 있어야 합니다. 여기서 백엔드 정의에는 클라이언트 인증서, 키, 신뢰할 수 있는 CA 인증서(사용되는 경우)의 Base64 인코딩 값이 포함되어야 합니다(권장).

기존 백엔드를 업데이트하여 자격 증명 기반 방식과 인증서 기반 방식 사이를 전환할 수 있습니다. 하지만 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

> *자격 증명과 인증서*를 모두 제공하려고 하면 구성 파일에 두 개 이상의 인증 방법이 제공되었다는 오류와 함께 백엔드 생성이 실패합니다.

자격 증명 기반 인증 활성화

Trident ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. 다음과 같은 표준 사전 정의된 역할을 활용하는 것이 좋습니다. `admin` 또는 `vsadmin` . 이를 통해 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 공개할 수 있는 향후 ONTAP 릴리스와의 호환성이 보장됩니다. Trident 사용하면 사용자 정의 보안 로그인 역할을 만들어 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 곳이라는 점을 명심하세요. 백엔드가 생성된 후 사용자 이름/비밀번호는 Base64로 인코딩되어 Kubernetes 비밀로 저장됩니다. 백엔드를 생성/업데이트하는 단계는 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행해야 하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

새로운 백엔드와 기존 백엔드는 인증서를 사용하고 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 개의 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64로 인코딩된 값입니다.
- clientPrivateKey: 연관된 개인 키의 Base64 인코딩된 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않으면 무시할 수 있습니다.

일반적인 작업 흐름은 다음 단계로 구성됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성할 때, 인증할 ONTAP 사용자로 일반 이름(CN)을 설정합니다.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. ONTAP 클러스터에 신뢰할 수 있는 CA 인증서를 추가합니다. 이 문제는 이미 스토리지 관리자가 처리했을 수도 있습니다. 신뢰할 수 있는 CA가 사용되지 않으면 무시합니다.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. ONTAP 클러스터에 클라이언트 인증서와 키(1단계)를 설치합니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. ONTAP 보안 로그인 역할이 지원되는지 확인하세요. `cert` 인증방법.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 생성된 인증서를 사용하여 인증을 테스트합니다. < ONTAP 관리 LIF> 및 <vserver 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다. LIF의 서비스 정책이 다음과 같이 설정되어 있는지 확인해야 합니다. `default-data-management` .

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

인증 방법 업데이트 또는 자격 증명 순환

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 순환할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/비밀번호를 사용하는 백엔드는 인증서를 사용하도록 업데이트할 수 있고, 인증서를 활용하는 백엔드는 사용자 이름/비밀번호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새로운 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수를 포함하는 업데이트된 backend.json 파일을 사용하여 실행합니다. `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

비밀번호를 순환할 때 스토리지 관리자는 먼저 ONTAP 에서 사용자의 비밀번호를 업데이트해야 합니다. 이어서 백엔드 업데이트가 진행됩니다. 인증서를 순환할 때 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 그런 다음 백엔드를 업데이트하여 새 인증서를 사용하고, 그 후 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 만들어진 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공하면 Trident ONTAP 백엔드와 통신하고 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident 에 대한 사용자 정의 ONTAP 역할 생성

Trident 에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용하지 않아도 되도록 최소한의 권한으로 ONTAP 클러스터 역할을 만들 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident 사용자가 만든 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

참조하다"Trident 사용자 정의 역할 생성기" Trident 사용자 정의 역할 생성에 대한 자세한 내용은 다음을 참조하세요.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 만듭니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자에 대한 사용자 이름을 만듭니다.

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 사용자에게 역할을 매핑합니다.

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

시스템 관리자 사용

ONTAP 시스템 관리자에서 다음 단계를 수행합니다.

1. 사용자 정의 역할 만들기:
   a. 클러스터 수준에서 사용자 지정 역할을 만들려면 *클러스터 > 설정*을 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 만들려면 저장소 **>** 저장소 **VM > `required SVM` >** 설정 **>** 사용자 및 역할.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. *역할*에서 *+추가*를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. * Trident 사용자에게 역할 매핑*: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에 있는 추가 아이콘 *+*을 선택합니다.
   b. 필요한 사용자 이름을 선택하고, 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭하세요.

자세한 내용은 다음 페이지를 참조하세요.

- "ONTAP 관리를 위한 사용자 정의 역할"또는"사용자 정의 역할 정의"
- "역할 및 사용자 작업"

NFS 내보내기 정책 관리

Trident NFS 내보내기 정책을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어합니다.

Trident 수출 정책을 사용할 때 두 가지 옵션을 제공합니다.

- Trident 내보내기 정책을 직접 동적으로 관리할 수 있습니다. 이 작동 모드에서 스토리지 관리자는 허용 가능한 IP 주소를 나타내는 CIDR 블록 목록을 지정합니다. Trident 게시 시점에 해당 범위에 해당하는 노드 IP를 자동으로 내보내기 정책에 추가합니다. 또는 CIDR이 지정되지 않은 경우 볼륨이 게시되는 노드에서 발견된 모든 글로벌 범위 유니캐스트 IP가 내보내기 정책에 추가됩니다.
- 스토리지 관리자는 내보내기 정책을 만들고 규칙을 수동으로 추가할 수 있습니다. 구성에서 다른 내보내기 정책 이름이 지정되지 않는 한 Trident 기본 내보내기 정책을 사용합니다.

동적으로 수출 정책을 관리합니다

Trident ONTAP 백엔드에 대한 내보내기 정책을 동적으로 관리하는 기능을 제공합니다. 이를 통해 스토리지 관리자는 명시적 규칙을 수동으로 정의하는 대신, 작업자 노드 IP에 허용되는 주소 공간을 지정할 수 있습니다. 이를 통해 내보내기 정책 관리가 대폭 간소화됩니다. 내보내기 정책을 수정하더라도 더 이상 스토리지 클러스터에 대한 수동 개입이 필요하지 않습니다. 또한 이를 통해 볼륨을 마운트하고 지정된 범위 내의 IP를 가진 작업자 노드에만 스토리지 클러스터에 대한 액세스를 제한하여 세분화되고 자동화된 관리를 지원합니다.

> 동적 내보내기 정책을 사용할 때는 NAT(네트워크 주소 변환)를 사용하지 마세요. NAT를 사용하면 스토리지 컨트롤러는 실제 IP 호스트 주소가 아닌 프런트엔드 NAT 주소를 확인하므로 내보내기 규칙에서 일치하는 항목이 없으면 액세스가 거부됩니다.

예

사용해야 하는 구성 옵션은 두 가지가 있습니다. 백엔드 정의의 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

> 이 기능을 사용하려면 SVM의 루트 접합에 노드 CIDR 블록을 허용하는 내보내기 규칙(예: 기본 내보내기 정책)이 포함된 이전에 생성된 내보내기 정책이 있는지 확인해야 합니다. Trident 에 SVM을 전용으로 지정하려면 항상 NetApp 권장하는 모범 사례를 따르세요.

위의 예를 사용하여 이 기능이 작동하는 방식을 설명하겠습니다.

- `autoExportPolicy`로 설정됩니다 `true`. 이는 Trident 이 백엔드로 프로비저닝된 각 볼륨에 대한 내보내기 정책을 생성함을 나타냅니다. `svm1` SVM을 사용하여 규칙 추가 및 삭제를 처리합니다. `autoexportCIDRs` 주소 블록. 볼륨이 노드에 연결될 때까지 해당 볼륨은 원치 않는 볼륨 액세스를 방지하는 규칙이 없는 빈 내보내기 정책을 사용합니다. 볼륨이 노드에 게시되면 Trident 지정된 CIDR 블록 내의 노드 IP를 포함하는 기본 qtree와 동일한 이름으로 내보내기 정책을 생성합니다. 이러한 IP는 부모 FlexVol volume 에서 사용하는 내보내기 정책에도 추가됩니다.

◦ 예를 들어:

- 백엔드 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
- `autoExportPolicy`로 설정 `true`
- 저장 접두사 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c라는 qtree는 FlexVol 에 대한 내보내기 정책을 생성합니다. `trident-403b5326-8482-40db96d0-d83fb3f4daec` , qtree에 대한 내보내기 정책 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` , 그리고 이름이 비어 있는 내보내기 정책 `trident_empty` SVM에 대해. FlexVol 내보내기 정책에 대한 규칙은 qtree 내보내기 정책에 포함된 모든 규칙의 상위 집합입니다. 빈 내보내기 정책은 첨부되지 않은 모든 볼륨에서 재사용됩니다.

- `autoExportCIDRs`주소 블록 목록이 포함되어 있습니다. 이 필드는 선택 사항이며 기본값은 ["0.0.0.0/0", "::/0"]입니다. 정의되지 않은 경우, Trident 작업자 노드에서 발견된 모든 글로벌 범위 유니캐스트 주소를 게시물과 함께 추가합니다.

이 예에서는 `192.168.0.0/24` 주소 공간이 제공됩니다. 이는 게시된 내용이 있는 이 주소 범위에 속하는 Kubernetes 노드 IP가 Trident 에서 생성하는 내보내기 정책에 추가됨을 나타냅니다. Trident 실행되는 노드를 등록하면 노드의 IP 주소를 검색하여 제공된 주소 블록과 비교합니다. `autoExportCIDRs` 게시 시점에 IP를 필터링한 후 Trident 게시 대상 노드의 클라이언트 IP에 대한 내보내기 정책 규칙을 만듭니다.

업데이트할 수 있습니다 `autoExportPolicy` 그리고 `autoExportCIDRs` 백엔드를 만든 후 백엔드에 대해 알아보세요. 자동으로 관리되는 백엔드에 새로운 CIDR을 추가하거나 기존 CIDR을 삭제할 수 있습니다. CIDR을 삭제할 때는 기존 연결이 끊어지지 않도록 주의하세요. 비활성화하도록 선택할 수도 있습니다. `autoExportPolicy` 백엔드의 경우 수동으로 만든 내보내기 정책으로 돌아갑니다. 이렇게 하려면 다음을 설정해야 합니다. `exportPolicy` 백엔드 구성의 매개변수입니다.

Trident 백엔드를 생성하거나 업데이트한 후에는 다음을 사용하여 백엔드를 확인할 수 있습니다. `tridentctl` 또는 해당 `tridentbackend` CRD:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

노드가 제거되면 Trident 모든 내보내기 정책을 확인하여 노드에 해당하는 액세스 규칙을 제거합니다. Trident 관리형 백엔드의 내보내기 정책에서 이 노드 IP를 제거하여 클러스터의 새 노드에서 이 IP를 재사용하지 않는 한 악성 마운트를 방지합니다.

기존 백엔드의 경우 백엔드를 업데이트합니다. `tridentctl update backend` Trident 자동으로 수출 정책을 관리하도록 보장합니다. 이렇게 하면 필요할 때 백엔드의 UUID와 qtree 이름을 따서 명명된 두 개의 새로운 내보내기 정책이 생성됩니다. 백엔드에 있는 볼륨은 마운트 해제 후 다시 마운트되면 새로 생성된 내보내기 정책을 사용합니다.

> ⓘ 자동 관리형 내보내기 정책이 있는 백엔드를 삭제하면 동적으로 생성된 내보내기 정책도 삭제됩니다. 백엔드를 다시 만들면 새로운 백엔드로 처리되어 새로운 내보내기 정책이 생성됩니다.

라이브 노드의 IP 주소가 업데이트되면 노드에서 Trident Pod를 다시 시작해야 합니다. 그러면 Trident 해당 IP 변경 사항을 반영하기 위해 관리하는 백엔드에 대한 내보내기 정책을 업데이트합니다.

SMB 볼륨 프로비저닝 준비

약간의 추가 준비로 다음을 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. `ontap-nas` 운전자.

> ⓘ SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성하여 다음을 생성해야 합니다. `ontap-nas-economy` ONTAP 온프레미스 클러스터를 위한 SMB 볼륨. 이러한 프로토콜 중 하나라도 구성하지 못하면 SMB 볼륨 생성이 실패합니다.

> ⓘ `autoExportPolicy`SMB 볼륨에서는 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하려면 다음이 필요합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명을 포함하는 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

단계

1. 온프레미스 ONTAP 의 경우 선택적으로 SMB 공유를 만들 수 있으며, Trident 만들어줄 수도 있습니다.

   Amazon FSx for ONTAP 에는 SMB 공유가 필요합니다.

   다음 두 가지 방법 중 하나를 사용하여 SMB 관리자 공유를 생성할 수 있습니다."Microsoft 관리 콘솔" 공유 폴더 스냅인 또는 ONTAP CLI 사용. ONTAP CLI를 사용하여 SMB 공유를 생성하려면:

   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 만듭니다.

      그만큼 `vserver cifs share create` 이 명령은 공유 생성 중에 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연관된 SMB 공유를 만듭니다.

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. 공유가 생성되었는지 확인하세요.

```
vserver cifs share show -share-name share_name
```

   참조하다"SMB 공유 만들기" 자세한 내용은 다음을 참조하세요.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대해서는 다음을 참조하세요."FSx for ONTAP 구성 옵션 및 예제" .

매개변수	설명	예
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름, Trident SMB 공유를 만들 수 있도록 하는 이름, 또는 매개변수를 비워 두면 볼륨에 대한 일반 공유 액세스를 방지할 수 있습니다. 이 매개변수는 온프레미스 ONTAP 의 경우 선택 사항입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필수이므로 비워둘 수 없습니다.	smb-share
nasType	설정해야 합니다 **smb .** null인 경우 기본값은 다음과 같습니다. nfs .	smb
securityStyle	새로운 볼륨에 대한 보안 스타일입니다. 설정해야 합니다 **ntfs** 또는 **mixed SMB** 볼륨의 경우**.**	ntfs`또는 `mixed SMB 볼륨의 경우
unixPermissions	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워두어야 합니다**.**	""

보안 **SMB** 활성화

25.06 릴리스부터 NetApp Trident 다음을 사용하여 생성된 SMB 볼륨의 보안 프로비저닝을 지원합니다. ontap-nas 그리고 ontap-nas-economy 백엔드. 보안 SMB가 활성화되면 액세스 제어 목록(ACL)을 사용하여 Active Directory(AD) 사용자 및 사용자 그룹의 공유에 대한 SMB에 대한 제어된 액세스를 제공할 수 있습니다.

기억해야 할 점

- 수입 ontap-nas-economy 볼륨은 지원되지 않습니다.
- 읽기 전용 복제본만 지원됩니다. ontap-nas-economy 볼륨.
- Secure SMB가 활성화된 경우 Trident 백엔드에 언급된 SMB 공유를 무시합니다.
- PVC 주석, 스토리지 클래스 주석 및 백엔드 필드를 업데이트해도 SMB 공유 ACL은 업데이트되지 않습니다.
- 복제 PVC의 주석에 지정된 SMB 공유 ACL은 소스 PVC의 ACL보다 우선합니다.
- 보안 SMB를 활성화하는 동안 유효한 AD 사용자를 제공해야 합니다. 유효하지 않은 사용자는 ACL에 추가되지 않습니다.
- 백엔드, 스토리지 클래스, PVC에서 동일한 AD 사용자에게 서로 다른 권한을 제공하는 경우 권한 우선순위는 PVC, 스토리지 클래스, 백엔드 순입니다.
- 보안 SMB가 지원됩니다. ontap-nas 관리되는 볼륨 가져오기에는 적용되며 관리되지 않는 볼륨 가져오기에는 적용되지 않습니다.

단계

1. 다음 예와 같이 TridentBackendConfig에 adAdminUser를 지정합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 저장 클래스에 주석을 추가합니다.

   추가하세요 `trident.netapp.io/smbShareAdUser` 실패 없이 안전한 SMB를 활성화하기 위해 저장 클래스에 주석을 추가합니다. 주석에 지정된 사용자 값 `trident.netapp.io/smbShareAdUser` 사용자 이름과 동일해야 합니다. `smbcreds` 비밀. 다음 중 하나를 선택할 수 있습니다. `smbShareAdUserPermission` : `full_control` , `change` , 또는 `read` . 기본 권한은 다음과 같습니다. `full_control` .

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. PVC를 만듭니다.

   다음 예제에서는 PVC를 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS 구성 옵션 및 예

Trident 설치로 ONTAP NAS 드라이버를 만들고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClass에 매핑하기 위한 세부 정보를 제공합니다.

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	기본
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-nas`, `ontap-nas-economy` , 또는 `ontap-nas-flexgroup`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소, 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 원활한 MetroCluster 전환을 위해서는 다음을 참조하세요.MetroCluster 예제 .	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	기본
`dataLIF`	프로토콜 LIF의 IP 주소. NetApp 다음을 지정하는 것이 좋습니다. `dataLIF` . 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. 초기 설정 후 변경이 가능합니다. 참조하다 . IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 다음과 같이 대괄호로 정의해야 합니다. `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . 메트로클러스터는 생략합니다. 를 참조하십시오MetroCluster 예제 .	지정되지 않은 경우 지정된 주소 또는 SVM에서 파생됨(권장하지 않음)
`svm`	사용할 스토리지 가상 머신 **Metrocluster**의 경우 생략 를 참조하십시오MetroCluster 예제 .	SVM의 경우 파생됨 `managementLIF` 지정됨
`autoExportPolicy`	자동 내보내기 정책 생성 및 업데이트 활성화[부울] 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	거짓
`autoExportCIDRs`	Kubernetes 노드 IP를 필터링할 CIDR 목록 `autoExportPolicy` 활성화되어 있습니다. 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	["0.0.0.0/0", "::/0"]`
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됨	""
`username`	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	
`password`	클러스터/SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	

매개변수	설명	기본
`storagePrefix`	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 설정 후 업데이트 불가 ontap-nas-economy와 24자 이상의 storagePrefix를 사용하는 경우 볼륨 이름에는 스토리지 접두사가 포함되지만 qtree에는 포함되지 않습니다.	"삼지창"
`aggregate`	프로비저닝을 위한 집계(선택 사항, 설정된 경우 SVM에 할당해야 함). 를 위해 `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당되지 않은 경우 사용 가능한 모든 집계를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 집계가 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계를 구성한 경우, 집계의 이름이 변경되거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 SVM에 있는 집계로 변경하거나 집계를 완전히 제거해야 합니다.	""
`limitAggregateUsage`	사용량이 이 백분율을 초과하면 프로비저닝에 실패합니다. * Amazon FSx for ONTAP 에는 적용되지 않습니다*.	"" (기본적으로 적용되지 않음)
플렉스그룹집계목록	프로비저닝을 위한 집계 목록(선택 사항, 설정된 경우 SVM에 할당해야 함). SVM에 할당된 모든 집계는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다. **ontap-nas-flexgroup** 스토리지 드라이버에 대해 지원됩니다. SVM에서 집계 목록이 업데이트되면 Trident Controller를 다시 시작하지 않고도 SVM을 폴링하여 Trident 에서 목록이 자동으로 업데이트됩니다. Trident 에서 볼륨을 프로비저닝하기 위해 특정 집계 목록을 구성한 경우, 집계 목록의 이름이 바뀌거나 SVM 외부로 이동하면 SVM 집계를 폴링하는 동안 백엔드가 Trident 에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 만들려면 집계 목록을 SVM에 있는 목록으로 변경하거나 집계 목록을 완전히 제거해야 합니다.	""

매개변수	설명	기본
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 qtree에 대해 관리하는 볼륨의 최대 크기를 제한합니다. qtreesPerFlexvol 옵션을 사용하면 FlexVol volume 당 최대 Qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true} 사용하지 마십시오. debugTraceFlags 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.	널
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 nfs , smb 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
nfsMountOptions	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 지속형 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident 는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스나 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident 연관된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
qtreesPerFlexvol	FlexVol 당 최대 Qtree는 [50, 300] 범위 내에 있어야 합니다.	"200"
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름, Trident SMB 공유를 만들 수 있도록 하는 이름, 또는 매개변수를 비워 두면 볼륨에 대한 일반 공유 액세스를 방지할 수 있습니다. 이 매개변수는 온프레미스 ONTAP 의 경우 선택 사항입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필수이므로 비워둘 수 없습니다.	smb-share
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. useREST`설정 시 `true , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. false Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. ontapi 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. vsadmin 그리고 cluster-admin 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 useREST 로 설정됩니다 true 기본적으로; 변경 useREST 에게 false ONTAPI(ZAPI) 호출을 사용합니다.	true`ONTAP 9.15.1 이상인 경우, 그렇지 않은 경우 `false.
limitVolumePoolSize	ontap-nas-economy 백엔드에서 Qtrees를 사용할 때 요청 가능한 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본
denyNewVolumePools	제한하다 ontap-nas-economy 백엔드가 Qtree를 포함하기 위해 새로운 FlexVol 볼륨을 생성하지 못하도록 합니다. 새로운 PV를 프로비저닝하는 데는 기존 Flexvol만 사용됩니다.	
adAdminUser	SMB 공유에 대한 전체 액세스 권한이 있는 Active Directory 관리자 사용자 또는 사용자 그룹입니다. 이 매개변수를 사용하면 SMB 공유에 대한 전체 제어 권한을 관리자에게 제공할 수 있습니다.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. defaults 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
spaceAllocation	Qtrees에 대한 공간 할당	"진실"
spaceReserve	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움)	"없음"
snapshotPolicy	사용할 스냅샷 정책	"없음"
qosPolicy	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하세요.	""
adaptiveQosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. ontap-nas-economy에서 지원되지 않습니다.	""
snapshotReserve	스냅샷을 위해 예약된 볼륨의 백분율	"0"이면 snapshotPolicy "없음"이고, 그렇지 않으면 ""
splitOnClone	생성 시 부모로부터 복제본을 분할합니다.	"거짓"
encryption	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. false . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	"거짓"
tieringPolicy	"없음"을 사용하는 계층화 정책	
unixPermissions	새로운 볼륨에 대한 모드	NFS 볼륨의 경우 "777", SMB 볼륨의 경우 비어 있음(해당 없음)
snapshotDir	에 대한 액세스를 제어합니다. .snapshot 예배 규칙서	NFSv4의 경우 "true", NFSv3의 경우 "false"

매개변수	설명	기본
exportPolicy	사용할 수출 정책	"기본"
securityStyle	새로운 볼륨에 대한 보안 스타일입니다. NFS 지원 mixed 그리고 unix 보안 스타일. SMB 지원 mixed 그리고 ntfs 보안 스타일.	NFS 기본값은 unix . SMB 기본값은 ntfs .
nameTemplate	사용자 정의 볼륨 이름을 만드는 템플릿입니다.	""

Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되는지 확인해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.

볼륨 프로비저닝 예시

기본값이 정의된 예는 다음과 같습니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

을 위한 ontap-nas 그리고 ontap-nas-flexgroups Trident 이제 FlexVol snapshotReserve 백분율과 PVC에 맞게 올바르게 조정되도록 새로운 계산을 사용합니다. 사용자가 PVC를 요청하면 Trident 새로운 계산을 사용하여 더 많은 공간을 가진 원래 FlexVol 생성합니다. 이 계산은 사용자가 PVC에서 요청한 쓰기 가능 공간을 확보하고 요청한 것보다 적은 공간을 확보하지 않도록 보장합니다. v21.07 이전에는 사용자가 스냅샷 예약 비율을 50%로 설정한 PVC(예: 5GiB)를 요청하면 2.5GiB의 쓰기 가능 공간만 확보되었습니다. 이는 사용자가 요청한 것이 전체 볼륨이기

때문입니다. `snapshotReserve` 그 중 일부입니다. Trident 21.07을 사용하면 사용자가 요청하는 것은 쓰기 가능한 공간이며 Trident 이를 정의합니다. `snapshotReserve` 전체 볼륨에 대한 백분율로 나타낸 숫자입니다. 이것은 적용되지 않습니다 `ontap-nas-economy` . 작동 방식을 알아보려면 다음 예를 참조하세요.

계산은 다음과 같습니다.

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

에서 요청한 5GiB입니다. 그만큼 `volume show` 명령을 실행하면 다음 예와 비슷한 결과가 표시됩니다.

이전 설치의 기존 백엔드는 Trident 업그레이드 시 위에서 설명한 대로 볼륨을 프로비저닝합니다. 업그레이드 전에 생성한 볼륨의 경우, 변경 사항을 적용하려면 볼륨 크기를 조정해야 합니다. 예를 들어, 2GiB PVC `snapshotReserve=50` 이전에는 1GiB의 쓰기 가능 공간을 제공하는 볼륨이 생성되었습니다. 예를 들어 볼륨 크기를 3GiB로 조정하면 애플리케이션은 6GiB 볼륨에서 3GiB의 쓰기 가능 공간을 확보하게 됩니다.

최소 구성 예

다음 예에서는 대부분의 매개변수를 기본값으로 두는 기본 구성을 보여줍니다. 백엔드를 정의하는 가장 쉬운 방법입니다.

> Trident 와 함께 NetApp ONTAP 에서 Amazon FSx 사용하는 경우 IP 주소 대신 LIF에 DNS 이름을 지정하는 것이 좋습니다.

ONTAP NAS 경제 사례

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

ONTAP NAS Flexgroup 예시

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

MetroCluster 예제

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트하지 않아도 되도록 백엔드를 구성할 수 있습니다."SVM 복제 및 복구" .

원활한 전환 및 스위치백을 위해 다음을 사용하여 SVM을 지정하십시오. `managementLIF` 그리고 생략하다 `dataLIF` 그리고 `svm` 매개변수. 예를 들어:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

SMB 볼륨 예시

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

인증서 기반 인증 예제

이는 최소 백엔드 구성의 예입니다. `clientCertificate` , `clientPrivateKey` , 그리고 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)이 채워집니다. `backend.json` 그리고 클라이언트 인증서, 개인 키, 신뢰할 수 있는 CA 인증서의 base64 인코딩 값을 각각 가져옵니다.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

자동 내보내기 정책 예

이 예제에서는 Trident 동적 내보내기 정책을 사용하여 내보내기 정책을 자동으로 만들고 관리하는 방법을 보여줍니다. 이것은 다음과 같은 경우에도 동일하게 작동합니다. `ontap-nas-economy` 그리고 `ontap-nas-flexgroup` 운전자.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

IPv6 주소 예

이 예에서는 다음을 보여줍니다. `managementLIF` IPv6 주소를 사용합니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

SMB 볼륨을 사용하는 Amazon FSx for ONTAP 예제

그만큼 `smbShare` SMB 볼륨을 사용하는 FSx for ONTAP 에는 매개변수가 필요합니다.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

nameTemplate을 사용한 백엔드 구성 예

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

가상 풀을 사용한 백엔드의 예

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 다음과 같은 특정 기본값이 설정됩니다. `spaceReserve` 없음, `spaceAllocation` 거짓이고, `encryption` 거짓. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident "주석" 필드에 프로비저닝 라벨을 설정합니다. FlexVol 에 대한 의견이 설정되었습니다. `ontap-nas` 또는 FlexGroup 용 `ontap-nas-flexgroup` . Trident 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨에 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이러한 예에서 일부 스토리지 풀은 자체적으로 설정합니다. `spaceReserve` , `spaceAllocation` , 그리고 `encryption` 값이 지정되고 일부 풀은 기본값을 재정의합니다.

ONTAP NAS 예시

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
```

```
      app: wordpress
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

ONTAP NAS FlexGroup 예시

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
```

```
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

ONTAP NAS 경제 사례

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

백엔드를 StorageClass에 매핑

다음 StorageClass 정의는 다음을 참조합니다.가상 풀을 사용한 백엔드의 예 . 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

- 그만큼 `protection-gold` StorageClass는 첫 번째 및 두 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 골드 레벨 보호를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 그만큼 `protection-not-gold` StorageClass는 세 번째 및 네 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 이것들은 금 이외의 보호 수준을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 그만큼 `app-mysqldb` StorageClass는 네 번째 가상 풀에 매핑됩니다. `ontap-nas` 백엔드. 이는 mysqldb 유형 앱에 대한 스토리지 풀 구성을 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 그 `protection-silver-creditpoints-20k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-nas-flexgroup` 백엔드. 이 풀은 실버 레벨 보호와 20000 크레딧 포인트를 제공하는 유일한 풀입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 그만큼 `creditpoints-5k` StorageClass는 세 번째 가상 풀에 매핑됩니다. `ontap-nas` 백엔드와 두 번째 가상 풀 `ontap-nas-economy` 백엔드. 5000 크레딧 포인트를 제공하는 유일한 풀 서비스입니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident 어떤 가상 풀을 선택할지 결정하고 저장 요구 사항이 충족되는지 확인합니다.

업데이트 `dataLIF` 초기 구성 후

다음 명령을 실행하여 업데이트된 dataLIF로 새 백엔드 JSON 파일을 제공하면 초기 구성 후 dataLIF를 변경할 수 있습니다.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

PVC가 하나 이상의 포드에 연결된 경우 새로운 dataLIF가 적용되려면 해당 포드를 모두 내렸다가 다시 올려야 합니다.

보안 **SMB** 사례

ontap-nas 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas-economy 드라이버를 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

스토리지 풀을 사용한 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

ontap-nas 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

추가했는지 확인하세요 `annotations` 안전한 SMB를 구현합니다. 보안 SMB는 백엔드나 PVC에 설정된 구성과 관계없이 주석 없이는 작동하지 않습니다.

ontap-nas-economy 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

단일 **AD** 사용자가 있는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

여러 **AD** 사용자가 있는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Amazon FSx for NetApp ONTAP 과 함께 **Trident** 사용

"Amazon FSx for NetApp ONTAP"NetApp ONTAP 스토리지 운영 체제를 기반으로 하는 파일 시스템을 실행하고 실행할 수 있는 완전 관리형 AWS 서비스입니다. FSx for ONTAP 사용하면 AWS에 데이터를 저장함으로써 얻는 단순성, 민첩성, 보안성, 확장성의 이점을 누리는 동시에, 익숙한 NetApp 기능, 성능, 관리 역량을 활용할 수 있습니다. FSx for ONTAP ONTAP 파일 시스템 기능과 관리 API를 지원합니다.

Amazon FSx for NetApp ONTAP 파일 시스템을 Trident 와 통합하면 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터가 ONTAP 에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있습니다.

파일 시스템은 온프레미스 ONTAP 클러스터와 유사한 Amazon FSx 의 기본 리소스입니다. 각 SVM 내에서 하나 이상의 볼륨을 만들 수 있습니다. 볼륨은 파일 시스템의 파일과 폴더를 저장하는 데이터 컨테이너입니다. Amazon FSx for NetApp ONTAP 클라우드에서 관리형 파일 시스템으로 제공됩니다. 새로운 파일 시스템 유형은 * NetApp ONTAP*이라고 합니다.

Amazon FSx for NetApp ONTAP 과 함께 Trident 사용하면 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터가 ONTAP 에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있습니다.

요구 사항

또한"Trident 요구 사항" FSx for ONTAP Trident 와 통합하려면 다음이 필요합니다.

- 기존 Amazon EKS 클러스터 또는 자체 관리 Kubernetes 클러스터 `kubectl` 설치됨.
- 클러스터의 워커 노드에서 접근할 수 있는 기존 Amazon FSx for NetApp ONTAP 파일 시스템 및 스토리지 가상 머신(SVM)입니다.
- 준비된 작업자 노드"NFS 또는 iSCSI" .

> Amazon Linux 및 Ubuntu에 필요한 노드 준비 단계를 따르세요. "Amazon Machine Images" (AMI)는 EKS AMI 유형에 따라 다릅니다.

고려 사항

- SMB 볼륨:
  - SMB 볼륨은 다음을 사용하여 지원됩니다. `ontap-nas` 운전자만.
  - SMB 볼륨은 Trident EKS 애드온에서 지원되지 않습니다.
  - Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 참조하다 "SMB 볼륨 프로비저닝 준비" 자세한 내용은.
- Trident 24.02 이전에는 자동 백업이 활성화된 Amazon FSx 파일 시스템에서 생성된 볼륨은 Trident 에서 삭제할 수 없었습니다. Trident 24.02 이상에서 이 문제를 방지하려면 다음을 지정하십시오. `fsxFilesystemID` , AWS `apiRegion` , AWS `apikey` , 그리고 AWS `secretKey` AWS FSx for ONTAP 의 백엔드 구성 파일에서.

> Trident 에 IAM 역할을 지정하는 경우 다음을 지정하지 않아도 됩니다. `apiRegion` , `apiKey` , 그리고 `secretKey` 필드를 Trident 에 명시적으로 지정합니다. 자세한 내용은 다음을 참조하세요 ."FSx for ONTAP 구성 옵션 및 예제" .

Trident SAN/iSCSI 및 **EBS-CSI** 드라이버 동시 사용

AWS(EKS, ROSA, EC2 또는 기타 인스턴스)에서 ontap-san 드라이버(예: iSCSI)를 사용하려는 경우 노드에 필요한 다중 경로 구성이 Amazon Elastic Block Store(EBS) CSI 드라이버와 충돌할 수 있습니다. 동일한 노드에 있는 EBS 디스크를 방해하지 않고 멀티패스 기능을 보장하려면 멀티패스 설정에서 EBS를 제외해야 합니다. 이 예에서는 다음을 보여줍니다. `multipath.conf` EBS 디스크를 다중 경로에서 제외하면서 필수 Trident 설정을 포함하는 파일:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

입증

Trident 두 가지 인증 모드를 제공합니다.

- 자격 증명 기반(권장): AWS Secrets Manager에 자격 증명을 안전하게 저장합니다. 당신은 사용할 수 있습니다 `fsxadmin` 파일 시스템 또는 사용자 `vsadmin` SVM에 맞게 사용자가 구성했습니다.

> Trident 다음과 같이 실행될 것으로 예상됩니다. `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름을 가진 사용자로 지정할 수 있습니다. Amazon FSx for NetApp ONTAP 다음이 있습니다. `fsxadmin` ONTAP 의 제한된 대체품인 사용자 `admin` 클러스터 사용자. 우리는 강력히 사용을 권장합니다 `vsadmin` Trident 와 함께.

- 인증서 기반: Trident SVM에 설치된 인증서를 사용하여 FSx 파일 시스템의 SVM과 통신합니다.

인증 활성화에 대한 자세한 내용은 드라이버 유형에 대한 인증을 참조하세요.

- "ONTAP NAS 인증"
- "ONTAP SAN 인증"

테스트된 **Amazon Machine Images(AMI)**

EKS 클러스터는 다양한 운영 체제를 지원하지만 AWS는 컨테이너와 EKS에 맞게 특정 Amazon Machine Image(AMI)를 최적화했습니다. 다음 AMI는 NetApp Trident 25.02에서 테스트되었습니다.

아미	NAS	NAS-경제	iSCSI	iSCSI 경제
AL2023_x86_64_STANDARD	예	예	예	예
AL2_x86_64	예	예	예*	예*
BOTTLEROCKET_x86_64	예**	예	해당 없음	해당 없음
AL2023_ARM_64_STANDARD	예	예	예	예
AL2_ARM_64	예	예	예*	예*
BOTTLEROCKET_ARM_64	예**	예	해당 없음	해당 없음

- * 노드를 재시작하지 않고는 PV를 삭제할 수 없습니다.
- ** Trident 버전 25.02에서는 NFSv3와 작동하지 않습니다.

> 원하는 AMI가 여기에 나열되어 있지 않더라도 지원되지 않는다는 의미는 아닙니다. 단순히 테스트를 거치지 않았다는 의미일 뿐입니다. 이 목록은 AMI가 작동하는 것으로 알려진 방법에 대한 가이드 역할을 합니다.

다음과 함께 수행된 테스트**:**

- EKS 버전: 1.32
- 설치 방법: Helm 25.06 및 AWS 추가 기능 25.06
- NAS의 경우 NFSv3와 NFSv4.1이 모두 테스트되었습니다.
- SAN의 경우 iSCSI만 테스트되었으며 NVMe-oF는 테스트되지 않았습니다.

수행된 테스트:

- 생성: 스토리지 클래스, pvc, pod
- 삭제: pod, pvc(일반, qtree/lun – 경제형, AWS 백업이 있는 NAS)

더 많은 정보를 찾아보세요

- "Amazon FSx for NetApp ONTAP 설명서"
- "Amazon FSx for NetApp ONTAP 에 대한 블로그 게시물"

IAM 역할 및 AWS Secret 생성

명시적인 AWS 자격 증명을 제공하는 대신 AWS IAM 역할로 인증하여 Kubernetes 포드가 AWS 리소스에 액세스하도록 구성할 수 있습니다.

> AWS IAM 역할을 사용하여 인증하려면 EKS를 사용하여 배포된 Kubernetes 클러스터가 있어야 합니다.

AWS Secrets Manager 비밀 만들기

Trident 스토리지를 관리하기 위해 FSx vserver에 대한 API를 발행하므로 이를 위해서는 자격 증명이 필요합니다. 이러한 자격 증명을 전달하는 안전한 방법은 AWS Secrets Manager 비밀을 사용하는 것입니다. 따라서 아직 없으면 vsadmin 계정의 자격 증명이 포함된 AWS Secrets Manager 비밀을 만들어야 합니다.

이 예제에서는 Trident CSI 자격 증명을 저장하기 위해 AWS Secrets Manager 비밀을 만듭니다.

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

IAM 정책 생성

Trident 도 올바르게 실행하려면 AWS 권한이 필요합니다. 따라서 Trident 필요한 권한을 부여하는 정책을 만들어야 합니다.

다음 예제에서는 AWS CLI를 사용하여 IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

정책 **JSON** 예시:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

서비스 계정 연결(IRSA)을 위한 Pod ID 또는 IAM 역할 생성

Kubernetes 서비스 계정을 구성하여 EKS Pod Identity 또는 서비스 계정 연결(IRSA)을 위한 IAM 역할을 통해 AWS Identity and Access Management(IAM) 역할을 맡을 수 있습니다. 서비스 계정을 사용하도록 구성된 모든 Pod는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

포드 아이덴티티

Amazon EKS Pod Identity 연결은 Amazon EC2 인스턴스 프로필이 Amazon EC2 인스턴스에 자격 증명을 제공하는 방식과 유사하게 애플리케이션의 자격 증명을 관리하는 기능을 제공합니다.

EKS 클러스터에 **Pod Identity** 설치:

AWS 콘솔을 통해 또는 다음 AWS CLI 명령을 사용하여 Pod ID를 생성할 수 있습니다.

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

자세한 내용은 다음을 참조하세요."Amazon EKS Pod Identity Agent 설정" .

trust-relationship.json 생성:

EKS 서비스 주체가 Pod Identity에 대한 이 역할을 수행할 수 있도록 trust-relationship.json을 생성합니다. 그런 다음 이 신뢰 정책으로 역할을 만듭니다.

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json 파일:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

IAM 역할에 역할 정책 첨부:

이전 단계의 역할 정책을 생성된 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

포드 **ID** 연결 생성:

IAM 역할과 Trident 서비스 계정(trident-controller) 간에 Pod ID 연결을 만듭니다.

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

서비스 계정 연결**(IRSA)**을 위한 **IAM** 역할

AWS CLI 사용:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json 파일**:**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

다음 값을 업데이트하세요. `trust-relationship.json` 파일:

- **<account_id>** - AWS 계정 ID
- **<oidc_provider>** - EKS 클러스터의 OIDC입니다. 다음을 실행하여 oidc_provider를 얻을 수 있습니다.

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

IAM 정책을 사용하여 **IAM** 역할 연결:

역할이 생성되면 다음 명령을 사용하여 위 단계에서 생성된 정책을 역할에 연결합니다.

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

OICD 공급자가 연결되어 있는지 확인하세요:

OIDC 공급자가 클러스터와 연결되어 있는지 확인하세요. 다음 명령을 사용하여 확인할 수 있습니다.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

출력이 비어 있으면 다음 명령을 사용하여 IAM OIDC를 클러스터에 연결합니다.

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

eksctl을 사용하는 경우 다음 예를 사용하여 EKS의 서비스 계정에 대한 IAM 역할을 생성하세요.

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Trident 설치

Trident Kubernetes에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다.

다음 방법 중 하나를 사용하여 Trident 설치할 수 있습니다.

- 지배
- EKS 애드온

스냅샷 기능을 활용하려면 CSI 스냅샷 컨트롤러 애드온을 설치하세요. 참조하다"CSI 볼륨에 대한 스냅샷 기능 활성화" 자세한 내용은.

helm을 통해 **Trident** 설치

포드 아이덴티티

1. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 다음 예를 사용하여 Trident 설치하세요.

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

당신은 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등의 설치 세부 정보를 검토하는 명령입니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

서비스 계정 연결**(IRSA)**

1. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 클라우드 공급자 및 *클라우드 ID*에 대한 값을 설정합니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

당신은 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전, 개정 번호 등의 설치 세부 정보를 검토하는 명령입니다.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2506.0    25.06.0
```

iSCSI를 사용하려면 클라이언트 컴퓨터에서 iSCSI가 활성화되어 있는지 확인하세요. AL2023 Worker 노드 OS를 사용하는 경우 helm 설치에서 node prep 매개변수를 추가하여 iSCSI 클라이언트 설치를 자동화할 수 있습니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

EKS 애드온을 통해 **Trident** 설치

Trident EKS 애드온에는 최신 보안 패치와 버그 수정이 포함되어 있으며, AWS에서 Amazon EKS와 함께 작동하도록 검증되었습니다. EKS 추가 기능을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장할 수 있으며 추가 기능을 설치, 구성, 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

필수 조건

AWS EKS에 대한 Trident 추가 기능을 구성하기 전에 다음 사항이 있는지 확인하세요.

- 추가 구독이 있는 Amazon EKS 클러스터 계정
- AWS 마켓플레이스에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

AWS에 **Trident** 애드온 활성화

관리 콘솔

1. Amazon EKS 콘솔을 엽니다. https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 구성하려는 클러스터의 이름을 선택합니다.
4. *추가 기능*을 선택한 다음 *더 많은 추가 기능 받기*를 선택하세요.
5. 추가 기능을 선택하려면 다음 단계를 따르세요.
   a. **AWS Marketplace** 추가 기능 섹션까지 아래로 스크롤하여 검색 상자에 *"Trident"*를 입력합니다.
   b. NetApp 의 Trident 상자 오른쪽 상단에 있는 확인란을 선택하세요.
   c. *다음*을 선택하세요.
6. 선택한 추가 기능 구성 설정 페이지에서 다음을 수행합니다.

   Pod Identity 연결을 사용하는 경우 이 단계를 건너뜁니다.

   a. 사용하고 싶은 *버전*을 선택하세요.
   b. IRSA 인증을 사용하는 경우 선택적 구성 설정에서 사용 가능한 구성 값을 설정해야 합니다.
      - 사용하고 싶은 *버전*을 선택하세요.
      - 추가 기능 구성 스키마*를 따르고 *구성 값 섹션의 **configurationValues** 매개변수를 이전 단계에서 만든 role-arn으로 설정합니다(값은 다음 형식이어야 함).

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
충돌 해결 방법에 대해 재정의를 선택하면 기존 추가 기능에 대한 하나 이상의 설정을 Amazon EKS 추가 기능 설정으로 덮어쓸 수 있습니다. 이 옵션을 활성화하지 않고 기존 설정과 충돌이 발생하면 작업이 실패합니다. 발생한 오류 메시지를 사용하여 충돌 문제를 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 사용자가 직접 관리해야 하는 설정을 관리하지 않는지 확인하세요.

7. *다음*을 선택하세요.
8. 검토 및 추가 페이지에서 *만들기*를 선택하세요.

   애드온 설치가 완료되면 설치된 애드온이 표시됩니다.

AWS CLI

1. 생성하다 **`add-on.json`** 파일:

Pod Identity의 경우 다음 형식을 사용하세요:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

IRSA 인증의 경우 다음 형식을 사용하세요:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

바꾸다 `<role ARN>` 이전 단계에서 생성된 역할의 ARN을 사용합니다.

2. Trident EKS 애드온을 설치하세요.

```
aws eks create-addon --cli-input-json file://add-on.json
```

엑시틀

다음 예제 명령은 Trident EKS 추가 기능을 설치합니다.

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Trident EKS 애드온 업데이트

관리 콘솔

1. Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 업데이트하려는 클러스터의 이름을 선택합니다.
4. 추가 기능 탭을 선택하세요.
5. * NetApp 의 Trident *를 선택한 다음 *편집*을 선택합니다.
6. * NetApp 의 Trident 구성* 페이지에서 다음을 수행합니다.
   a. 사용하고 싶은 *버전*을 선택하세요.
   b. *선택적 구성 설정*을 확장하고 필요에 따라 수정합니다.
   c. *변경 사항 저장*을 선택하세요.

AWS CLI

다음 예제에서는 EKS 추가 기능을 업데이트합니다.

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

엑시틀

- FSxN Trident CSI 애드온의 현재 버전을 확인하세요. 바꾸다 `my-cluster` 클러스터 이름으로.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  예시 출력:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 이전 단계의 출력에서 UPDATE AVAILABLE에 반환된 버전으로 애드온을 업데이트합니다.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

제거하면 `--force` 옵션과 Amazon EKS 추가 기능 설정이 기존 설정과 충돌하는 경우 Amazon EKS 추가 기능 업데이트가 실패합니다. 충돌을 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 사용자가 관리해야 하는 설정을 관리하지 않는지 확인하세요. 이 옵션을 사용하면 해당 설정이 덮어쓰기되기 때문입니다. 이 설정에 대한 다른 옵션에 대한 자세한 내용은 다음을 참조하세요."애드온" . Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 다음을 참조하세요."쿠버네티스 필드 관리" .

Trident EKS 애드온 제거/제거

Amazon EKS 추가 기능을 제거하는 데는 두 가지 옵션이 있습니다.

- 클러스터에 추가 소프트웨어 유지 – 이 옵션을 선택하면 Amazon EKS에서 모든 설정을 관리하지 않습니다. 또한 Amazon EKS에서 업데이트를 알리고, 업데이트를 시작한 후 Amazon EKS 추가 기능을 자동으로 업데이트하는 기능도 제거됩니다. 하지만 클러스터의 추가 소프트웨어는 그대로 유지됩니다. 이 옵션을 선택하면 추가 기능이 Amazon EKS 추가 기능이 아닌 자체 관리형 설치가 됩니다. 이 옵션을 사용하면 추가 기능에 다운타임이 발생하지 않습니다. 유지하다 `--preserve` 추가 기능을 보존하려면 명령에 옵션을 추가하세요.
- 클러스터에서 애드온 소프트웨어를 완전히 제거합니다 – NetApp 클러스터에 종속된 리소스가 없는 경우에만 클러스터에서 Amazon EKS 애드온을 제거할 것을 권장합니다. 제거하다 `--preserve` 옵션에서 `delete` 추가 기능을 제거하는 명령입니다.

추가 기능에 IAM 계정이 연결되어 있는 경우 해당 IAM 계정은 제거되지 않습니다.

관리 콘솔

1. Amazon EKS 콘솔을 엽니다. https://console.aws.amazon.com/eks/home#/clusters .
2. 왼쪽 탐색 창에서 *클러스터*를 선택합니다.
3. NetApp Trident CSI 추가 기능을 제거할 클러스터의 이름을 선택합니다.
4. 추가 기능 탭을 선택한 다음 * Trident by NetApp*을 선택합니다.*
5. *제거*를 선택하세요.
6. **netapp_trident-operator** 제거 확인 대화 상자에서 다음을 수행합니다.
   a. Amazon EKS가 애드온에 대한 설정 관리를 중지하도록 하려면 *클러스터에 유지*를 선택합니다. 클러스터에 애드온 소프트웨어를 유지하여 애드온의 모든 설정을 직접 관리하려는 경우 이렇게 하세요.
   b. *netapp_trident-operator*를 입력합니다.
   c. *제거*를 선택하세요.

AWS CLI

바꾸다 `my-cluster` 클러스터 이름을 입력한 후 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

엑시틀

다음 명령은 Trident EKS 애드온을 제거합니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

스토리지 백엔드 구성

ONTAP SAN 및 NAS 드라이버 통합

스토리지 백엔드를 만들려면 JSON 또는 YAML 형식으로 구성 파일을 만들어야 합니다. 이 파일에는 원하는 저장 유형(NAS 또는 SAN), 파일 시스템, 데이터를 가져올 SVM, 그리고 이를 통해 인증하는 방법을 지정해야 합니다. 다음 예제에서는 NAS 기반 스토리지를 정의하고 AWS 비밀번호를 사용하여 사용하려는 SVM에 자격 증명을 저장하는 방법을 보여줍니다.

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

다음 명령을 실행하여 Trident 백엔드 구성(TBC)을 만들고 검증합니다.

- yaml 파일에서 트라이던트 백엔드 구성(TBC)을 만들고 다음 명령을 실행합니다.

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- 트라이던트 백엔드 구성(TBC)이 성공적으로 생성되었는지 확인합니다.

```
Kubectl get tbc -n trident
```

```
NAME                           BACKEND NAME          BACKEND UUID
PHASE    STATUS

backend-tbc-ontap-nas          tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9    Bound    Success
```

FSx for ONTAP 드라이버 세부 정보

다음 드라이버를 사용하여 Trident Amazon FSx for NetApp ONTAP 과 통합할 수 있습니다.

- `ontap-san`: 프로비저닝된 각 PV는 자체 Amazon FSx for NetApp ONTAP 볼륨 내의 LUN입니다. 블록 저장에 권장됩니다.
- `ontap-nas`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 입니다. NFS 및 SMB에 권장됩니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 LUN 수가 있는 LUN입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 qtree이며, Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 qtree 수가 있습니다.
- `ontap-nas-flexgroup`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP FlexGroup 볼륨을 위한 전체 Amazon FSx입니다.

운전자 세부 정보는 다음을 참조하세요."NAS 드라이버" 그리고"SAN 드라이버" .

구성 파일이 생성되면 다음 명령을 실행하여 EKS 내에서 해당 파일을 생성합니다.

```
kubectl create -f configuration_file
```

상태를 확인하려면 다음 명령을 실행하세요.

```
kubectl get tbc -n trident
```

```
NAME                      BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

백엔드 고급 구성 및 예제

백엔드 구성 옵션은 다음 표를 참조하세요.

매개변수	설명	예
`version`		항상 1
`storageDriverName`	저장 드라이버의 이름	`ontap-nas`, `ontap-nas-economy` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy`
`backendName`	사용자 정의 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소, 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다. 당신이 제공하는 경우 `fsxFilesystemID` 아래에 `aws` 필드에서는 다음을 제공할 필요가 없습니다. `managementLIF` Trident SVM을 검색하기 때문에 `managementLIF` AWS에서 제공하는 정보입니다. 따라서 SVM(예: vsadmin)에서 사용자에 대한 자격 증명을 제공해야 하며 사용자에게 다음이 있어야 합니다. `vsadmin` 역할.	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	예
`dataLIF`	프로토콜 LIF의 IP 주소. * ONTAP NAS 드라이버*: NetApp dataLIF를 지정하는 것을 권장합니다. 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. 초기 설정 후 변경이 가능합니다. 참조하다 . * ONTAP SAN 드라이버*: iSCSI에 대해서는 지정하지 마세요. Trident ONTAP Selective LUN Map을 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCI LIF를 검색합니다. dataLIF가 명시적으로 정의된 경우 경고가 생성됩니다. IPv6 플래그를 사용하여 Trident 설치한 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다.	
`autoExportPolicy`	자동 내보내기 정책 생성 및 업데이트 활성화[부울] 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	`false`
`autoExportCIDRs`	Kubernetes 노드 IP를 필터링할 CIDR 목록 `autoExportPolicy` 활성화되어 있습니다. 를 사용하여 `autoExportPolicy` 그리고 `autoExportCIDRs` 옵션을 통해 Trident 자동으로 내보내기 정책을 관리할 수 있습니다.	"["0.0.0.0/0", "::/0"]"
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64로 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`clientPrivateKey`	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값입니다. 선택 과목. 인증서 기반 인증에 사용됩니다.	""

매개변수	설명	예
username	클러스터 또는 SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. 예를 들어, vsadmin.	
password	클러스터 또는 SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다.	
svm	사용할 스토리지 가상 머신	SVM managementLIF가 지정된 경우 파생됩니다.
storagePrefix	SVM에서 새로운 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 생성 후에는 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새로운 백엔드를 만들어야 합니다.	trident
limitAggregateUsage	* Amazon FSx for NetApp ONTAP에 대해서는 지정하지 마세요.* 제공된 fsxadmin 그리고 vsadmin Trident 사용하여 집계 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다.	사용하지 마세요.
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다. 또한 qtree 및 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다. qtreesPerFlexvol 옵션을 사용하면 FlexVol volume 당 최대 Qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
lunsPerFlexvol	Flexvol 볼륨당 최대 LUN은 [50, 200] 범위 내에 있어야 합니다. SAN만 해당.	“100”
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true} 사용하지 마십시오. debugTraceFlags 문제 해결을 위해 자세한 로그 덤프가 필요한 경우가 아니면요.	널
nfsMountOptions	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 지속형 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident 는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스나 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident 연관된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""

매개변수	설명	예
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 nfs , smb , 또는 null. 설정해야 합니다 **smb SMB** 볼륨의 경우. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
qtreesPerFlexvol	FlexVol volume 당 최대 Qtree는 [50, 300] 범위 내에 있어야 합니다.	"200"
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름 또는 Trident SMB 공유를 만들 수 있도록 하는 이름입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필요합니다.	smb-share
useREST	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. 설정 시 true Trident ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. ontap 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. vsadmin 그리고 cluster-admin 역할.	false
aws	AWS FSx for ONTAP 의 구성 파일에서 다음을 지정할 수 있습니다. fsxFilesystemID : AWS FSx 파일 시스템의 ID를 지정합니다. - apiRegion : AWS API 지역 이름. - apikey : AWS API 키. - secretKey : AWS 비밀 키.	"" "" ""
credentials	AWS Secrets Manager에 저장할 FSx SVM 자격 증명을 지정합니다. - name : SVM의 자격 증명을 포함하는 비밀의 Amazon 리소스 이름(ARN)입니다. - type : 설정 awsarn . 참조하다"AWS Secrets Manager 비밀 만들기" 자세한 내용은.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

다음 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. defaults 구성 섹션. 예를 들어, 아래의 구성 예를 참조하세요.

매개변수	설명	기본
`spaceAllocation`	LUN에 대한 공간 할당	`true`
`spaceReserve`	공간 예약 모드; "없음"(씬) 또는 "볼륨"(두꺼움)	`none`
`snapshotPolicy`	사용할 스냅샷 정책	`none`
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. Trident 에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용해야 하며, 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 작업 부하의 총 처리량에 대한 상한을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. ontap-nas-economy에서 지원되지 않습니다.	""
`snapshotReserve`	스냅샷에 예약된 볼륨의 백분율 "0"	만약에 `snapshotPolicy` ~이다 `none` , `else` ""
`splitOnClone`	생성 시 부모로부터 복제본을 분할합니다.	`false`
`encryption`	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. `false` . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	`false`
`luksEncryption`	LUKS 암호화를 활성화합니다. 참조하다"Linux Unified Key Setup(LUKS) 사용" . SAN만 해당.	""
`tieringPolicy`	사용할 계층화 정책 `none`	
`unixPermissions`	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워 두세요.	""

매개변수	설명	기본
`securityStyle`	새로운 볼륨에 대한 보안 스타일입니다. NFS 지원 `mixed` 그리고 `unix` 보안 스타일. SMB 지원 `mixed` 그리고 `ntfs` 보안 스타일.	NFS 기본값은 `unix` . SMB 기본값은 `ntfs` .

SMB 볼륨 프로비저닝 준비

다음을 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. `ontap-nas` 운전사. 완료하기 전에ONTAP SAN 및 NAS 드라이버 통합 다음 단계를 완료하세요.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음을 수행하십시오. `ontap-nas` 운전자는 다음 사항을 갖춰야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2019를 실행하는 하나 이상의 Windows 워커 노드가 있는 Kubernetes 클러스터입니다. Trident Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명을 포함하는 Trident 비밀이 하나 이상 있어야 합니다. 비밀을 생성하려면 `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Windows 서비스로 구성된 CSI 프록시. 구성하려면 `csi-proxy` , 참조하다"GitHub: CSI 프록시" 또는"GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우.

단계

1. SMB 주식을 생성합니다. 다음 두 가지 방법 중 하나를 사용하여 SMB 관리자 공유를 생성할 수 있습니다 ."Microsoft 관리 콘솔" 공유 폴더 스냅인 또는 ONTAP CLI 사용. ONTAP CLI를 사용하여 SMB 공유를 생성하려면:
   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 만듭니다.

      그만큼 `vserver cifs share create` 이 명령은 공유 생성 중에 -path 옵션에 지정된 경로를 확인합니다. 지정된 경로가 존재하지 않으면 명령이 실패합니다.

   b. 지정된 SVM과 연관된 SMB 공유를 만듭니다.

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. 공유가 생성되었는지 확인하세요.

```
vserver cifs share show -share-name share_name
```

      참조하다"SMB 공유 만들기" 자세한 내용은 다음을 참조하세요.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대해서는 다음을 참조하세요."FSx for ONTAP 구성 옵션 및 예제" .

매개변수	설명	예
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console이나 ONTAP CLI를 사용하여 만든 SMB 공유의 이름 또는 Trident SMB 공유를 만들 수 있도록 하는 이름입니다. 이 매개변수는 Amazon FSx for ONTAP 백엔드에 필요합니다.	`smb-share`
`nasType`	설정해야 합니다 **`smb`** . null인 경우 기본값은 다음과 같습니다. `nfs` .	`smb`
`securityStyle`	새로운 볼륨에 대한 보안 스타일입니다. 설정해야 합니다 **`ntfs`** 또는 **`mixed`** **SMB** 볼륨의 경우.	`ntfs`또는 `mixed` SMB 볼륨의 경우
`unixPermissions`	새로운 볼륨에 대한 모드입니다. **SMB** 볼륨의 경우 비워두어야 합니다.	""

스토리지 클래스 및 **PVC** 구성

Kubernetes StorageClass 객체를 구성하고 Trident 가 볼륨을 프로비저닝하는 방법을 지시하는 스토리지 클래스를 만듭니다. 구성된 Kubernetes StorageClass를 사용하여 PV에 대한 액세스를 요청하는 PersistentVolumeClaim(PVC)을 만듭니다. 그런 다음 PV를 포드에 장착할 수 있습니다.

스토리지 클래스 생성

Kubernetes StorageClass 객체 구성

그만큼 "Kubernetes StorageClass 객체" 객체는 Trident 해당 클래스에 사용되는 프로비저너로 식별하고 Trident 볼륨을 프로비저닝하는 방법을 지시합니다. NFS를 사용하여 볼륨에 대한 Storageclass를 설정하려면 이 예제를 사용하세요(전체 속성 목록은 아래의 Trident 속성 섹션을 참조하세요).

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

iSCSI를 사용하여 볼륨에 대한 Storageclass를 설정하려면 다음 예를 사용하세요.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

AWS Bottlerocket에서 NFSv3 볼륨을 프로비저닝하려면 필요한 항목을 추가하세요. `mountOptions` 저장 클래스로:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

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

스토리지 클래스 생성

단계

1. 이것은 Kubernetes 객체이므로 다음을 사용합니다. `kubectl` Kubernetes에서 생성하세요.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스를 볼 수 있어야 하며, Trident 가 백엔드에서 풀을 검색했어야 합니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

PVC를 만듭니다

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

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

PVC를 만든 후에는 볼륨을 포드에 마운트할 수 있습니다.

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

다음 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWX 접근이 가능한 **PVC**

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

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

iSCSI를 사용한 **PVC** 예제

이 예에서는 RWO 액세스가 있는 iSCSI용 기본 PVC를 보여줍니다. 이 PVC는 StorageClass와 연관되어 있습니다. `protection-gold` .

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

PVC 생성

단계

1. PVC을 생성합니다.

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

2. PVC 상태를 확인하세요.

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi       RWO                           5m
```

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

Trident 속성

이러한 매개변수는 주어진 유형의 볼륨을 프로비저닝하는 데 어떤 Trident 관리 스토리지 풀을 사용해야 하는지 결정합니다.

기인하다	유형	가치	권하다	요구	지원됨
미디어[1]	끈	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다 의미합니다.	지정된 미디어 유형	온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-산, 솔리드파이어-산
프로비저닝 유형	끈	얇은, 두꺼운	풀은 이 프로비저닝 방법을 지원합니다.	프로비저닝 방법이 지정됨	두꺼운: 모두 온탭; 얇은: 모두 온탭 & solidfire-san
백엔드 유형	끈	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	풀은 이 유형의 백엔드에 속합니다.	백엔드 지정됨	모든 운전자
스냅샷	부울	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다.	스냅샷이 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
클론	부울	참, 거짓	풀은 볼륨 복제를 지원합니다.	복제가 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
암호화	부울	참, 거짓	풀은 암호화된 볼륨을 지원합니다.	암호화가 활성화된 볼륨	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산

기인하다	유형	가치	권하다	요구	지원됨
아이옵스	정수	양의 정수	풀은 이 범위에서 IOPS를 보장할 수 있습니다.	볼륨은 이러한 IOPS를 보장합니다.	솔리드파이어-산

[1]: ONTAP Select 시스템에서는 지원되지 않습니다.

샘플 애플리케이션 배포

스토리지 클래스와 PVC가 생성되면 PV를 포드에 마운트할 수 있습니다. 이 섹션에서는 PV를 포드에 연결하는 명령과 구성의 예를 나열합니다.

단계

1. 볼륨을 포드에 마운트합니다.

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

다음 예에서는 PVC를 포드에 부착하기 위한 기본 구성을 보여줍니다. 기본 구성:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

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

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

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

```
Filesystem                                                                 Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

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

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

EKS 클러스터에서 **Trident EKS** 추가 기능 구성

NetApp Trident Kubernetes에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다. NetApp Trident EKS 애드온에는 최신 보안 패치와 버그 수정이 포함되어 있으며, AWS에서 Amazon EKS와 함께 작동하도록 검증되었습니다. EKS 추가 기능을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장할 수 있으며 추가 기능을 설치, 구성, 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

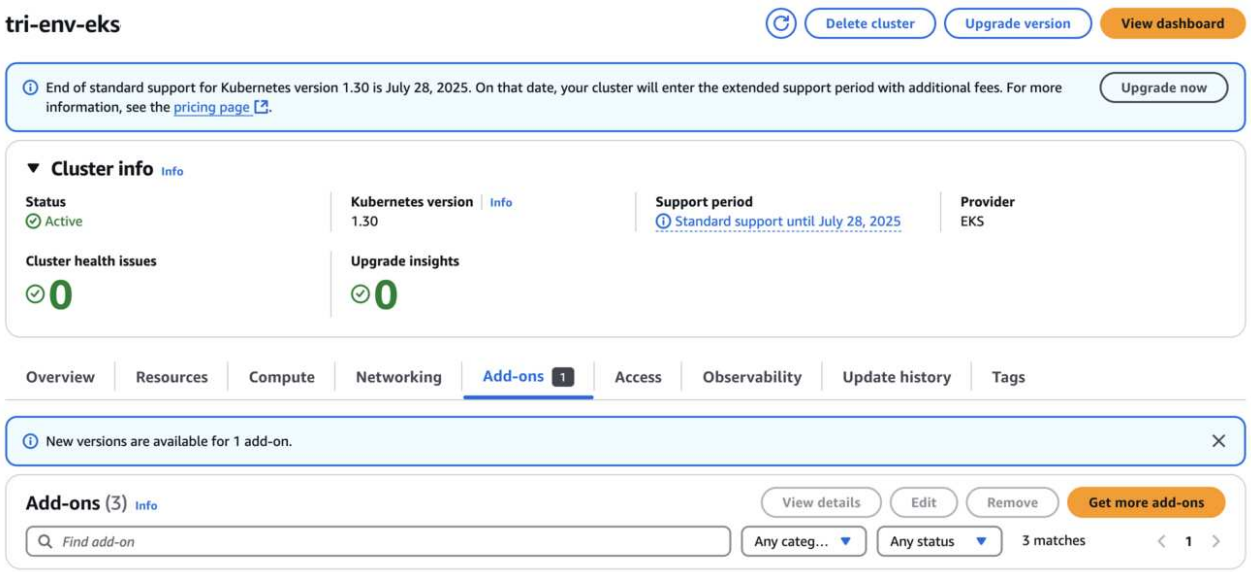
필수 조건

AWS EKS에 대한 Trident 추가 기능을 구성하기 전에 다음 사항이 있는지 확인하세요.

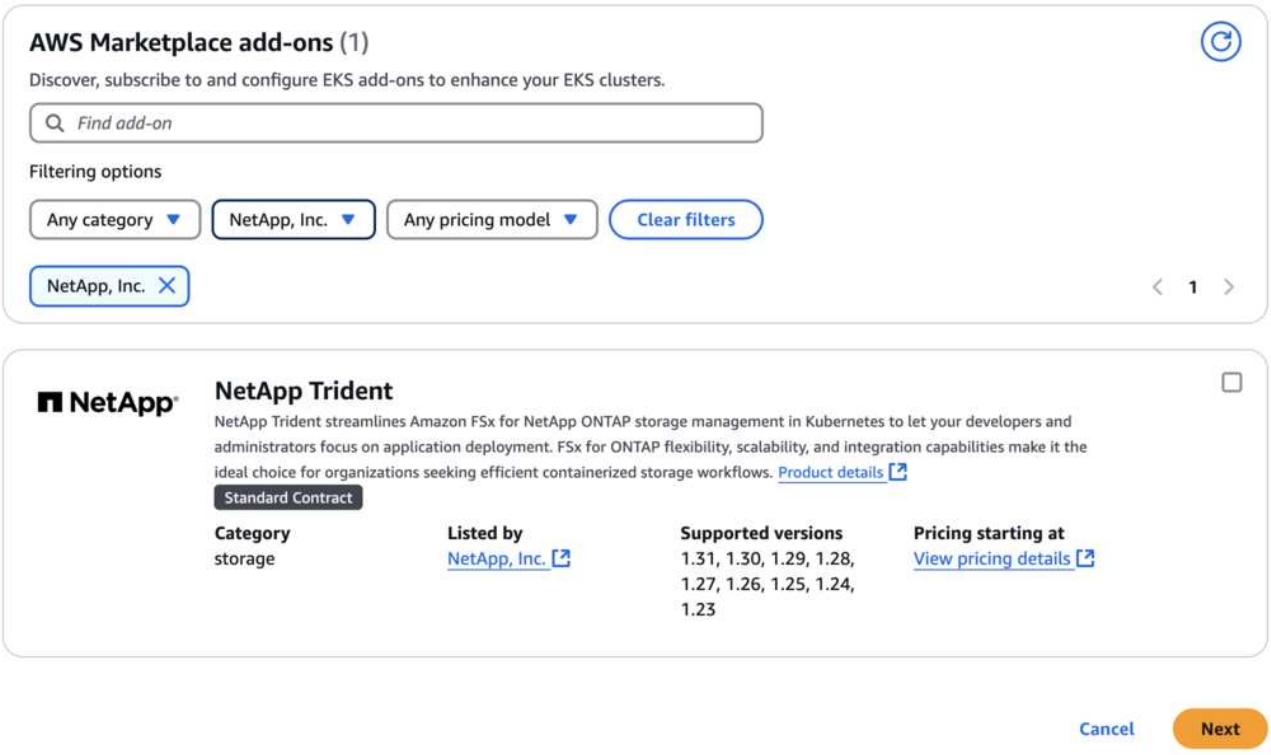
- 애드온을 사용할 수 있는 권한이 있는 Amazon EKS 클러스터 계정입니다. 참조하다"Amazon EKS 애드온" .
- AWS 마켓플레이스에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

단계

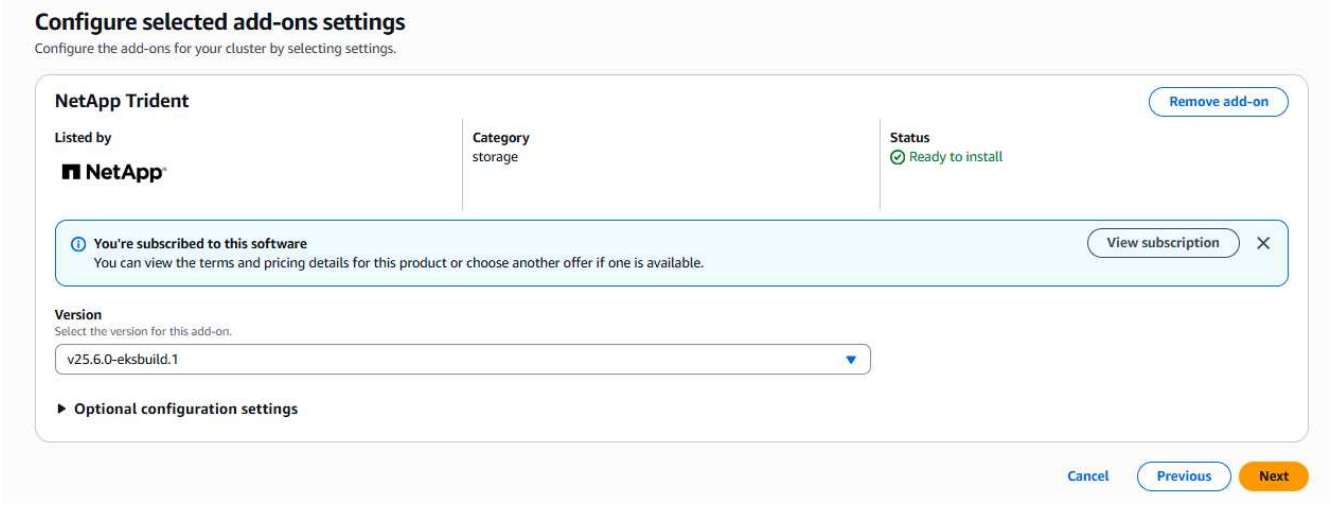
1. EKS Pod가 AWS 리소스에 액세스할 수 있도록 IAM 역할과 AWS 비밀번호를 생성해야 합니다. 지침은 다음을 참조하세요."IAM 역할 및 AWS Secret 생성" .
2. EKS Kubernetes 클러스터에서 추가 기능 탭으로 이동합니다.



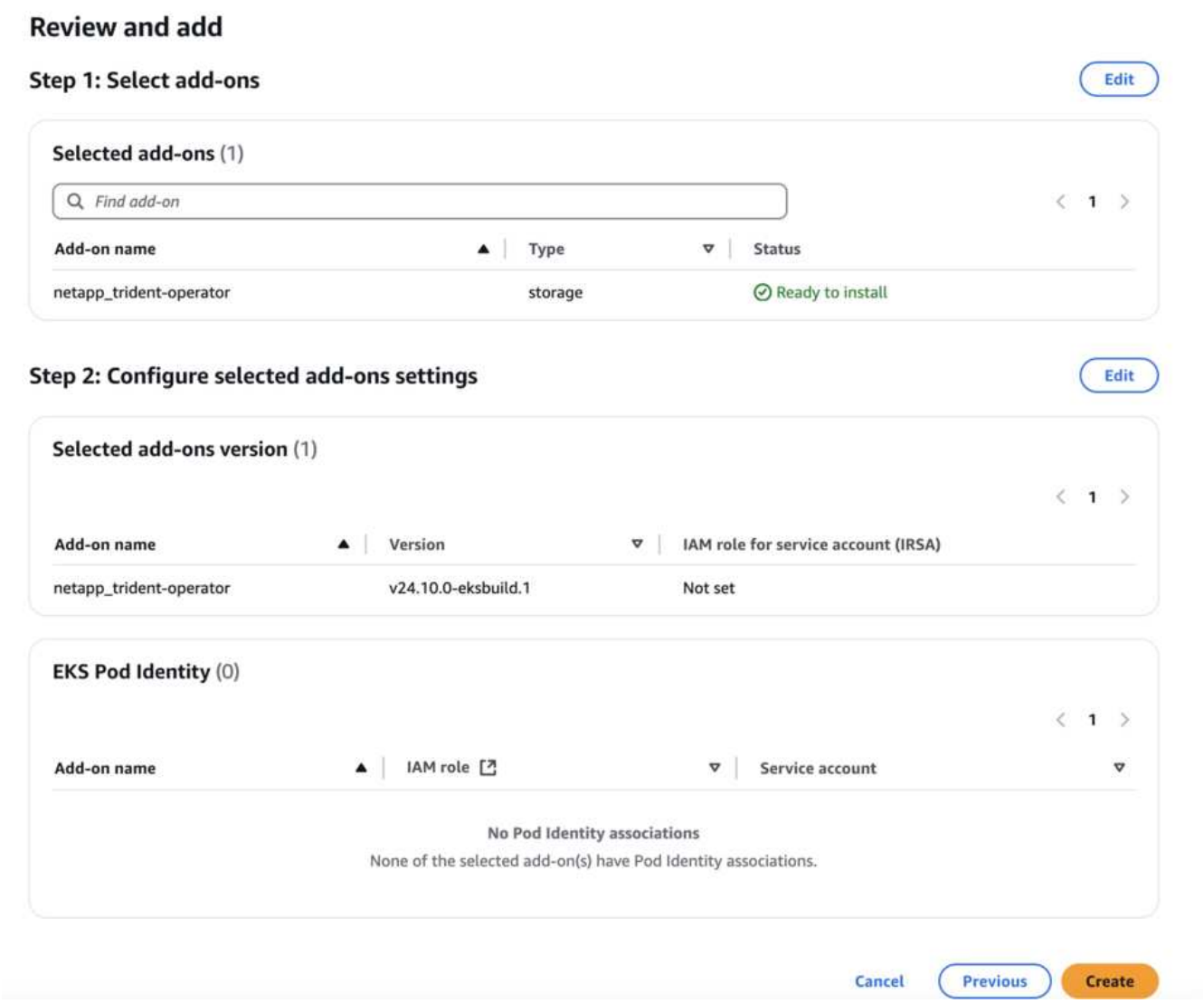
3. *AWS Marketplace 추가 기능*으로 이동하여 *storage* 카테고리를 선택합니다.



4. * NetApp Trident*를 찾아 Trident 추가 기능의 확인란을 선택하고 *다음*을 클릭합니다.
5. 원하는 애드온 버전을 선택하세요.

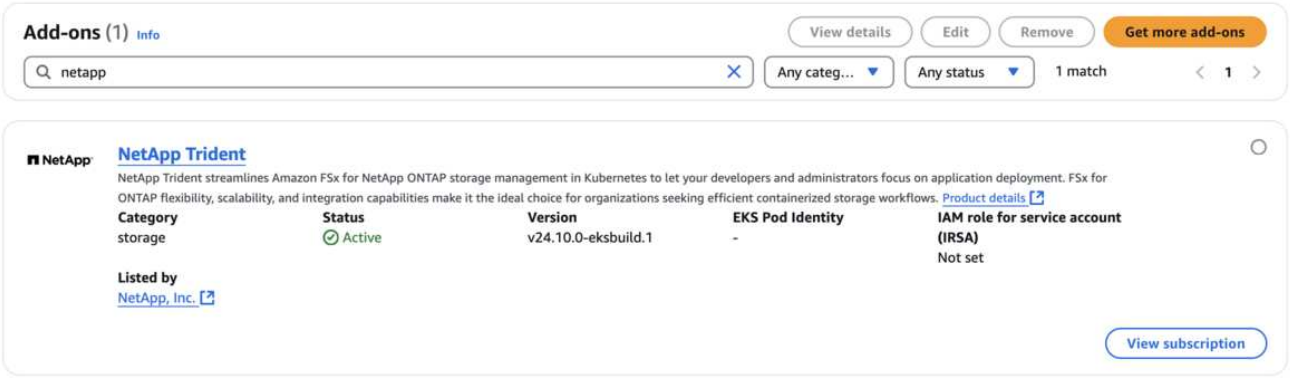


6. 필요한 추가 기능 설정을 구성합니다.



7. IRSA(서비스 계정에 대한 IAM 역할)를 사용하는 경우 추가 구성 단계를 참조하세요."여기" .
8. *만들기*를 선택하세요.

9. 추가 기능의 상태가 _활성_인지 확인하세요.



10. 다음 명령을 실행하여 Trident 클러스터에 제대로 설치되었는지 확인하세요.

```
kubectl get pods -n trident
```

11. 설정을 계속하고 스토리지 백엔드를 구성합니다. 자세한 내용은 다음을 참조하세요."스토리지 백엔드 구성" .

CLI를 사용하여 **Trident EKS** 애드온 설치/제거

CLI를 사용하여 **NetApp Trident EKS** 애드온을 설치하세요**.**

다음 예제 명령은 Trident EKS 추가 기능을 설치합니다.
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (전용 버전 포함)

CLI를 사용하여 **NetApp Trident EKS** 애드온을 제거합니다**.**

다음 명령은 Trident EKS 애드온을 제거합니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

kubectl로 백엔드 만들기

백엔드는 Trident 와 스토리지 시스템 간의 관계를 정의합니다. 이는 Trident 해당 스토리지 시스템과 통신하는 방법과 Trident 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Trident 설치한 후 다음 단계는 백엔드를 만드는 것입니다. 그만큼 `TridentBackendConfig` 사용자 정의 리소스 정의(CRD)를 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 만들고 관리할 수 있습니다. 다음을 사용하여 이 작업을 수행할 수 있습니다. `kubectl` 또는 Kubernetes 배포판에 맞는 동등한 CLI 도구입니다.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig` , `tbackendconfig`)는 Trident 백엔드를 관리할 수 있는 프런트엔드 네임스페이스 CRD입니다. `kubectl` . Kubernetes 및 스토리지 관리자는 이제 전용 명령줄 유틸리티가 필요 없이 Kubernetes CLI를 통해 직접 백엔드를 생성하고 관리할 수 있습니다.(`tridentctl`).

생성 시 `TridentBackendConfig` 객체의 경우 다음이 발생합니다.

- Trident 는 귀하가 제공한 구성을 기반으로 백엔드를 자동으로 생성합니다. 이는 내부적으로 다음과 같이 표현됩니다. `TridentBackend` (`tbe` , `tridentbackend`) CR.
- 그만큼 `TridentBackendConfig` 고유하게 결합됩니다 `TridentBackend` Trident 가 만든 것입니다.

각 `TridentBackendConfig` 일대일 매핑을 유지합니다. `TridentBackend` 전자는 사용자가 백엔드를 설계하고 구성하기 위해 제공하는 인터페이스이고, 후자는 Trident 실제 백엔드 객체를 표현하는 방식입니다.

> `TridentBackend`CR은 `Trident` 에 의해 자동으로 생성됩니다. 수정해서는 안 됩니다. 백엔드를 업데이트하려면 다음을 수정하세요. `TridentBackendConfig` 물체.

다음 예제를 참조하세요. `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

또한 다음 예제를 살펴볼 수도 있습니다. "트라이던트 설치 프로그램" 원하는 스토리지 플랫폼/서비스에 대한 샘플 구성을 위한 디렉토리입니다.

그만큼 `spec` 백엔드별 구성 매개변수를 사용합니다. 이 예에서 백엔드는 다음을 사용합니다. `ontap-san` 저장 드라이버이며 여기에 표로 정리된 구성 매개변수를 사용합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 다음을 참조하세요."스토리지 드라이버에 대한 백엔드 구성 정보" .

그만큼 `spec` 섹션에는 다음도 포함됩니다. `credentials` 그리고 `deletionPolicy` 새로 도입된 분야 `TridentBackendConfig` CR:

- `credentials`: 이 매개변수는 필수 필드이며 스토리지 시스템/서비스를 인증하는 데 사용되는 자격 증명을 포함합니다. 이는 사용자가 생성한 Kubernetes Secret으로 설정됩니다. 자격 증명은 일반 텍스트로 전달될 수 없으며 오류가 발생합니다.
- `deletionPolicy`: 이 필드는 다음과 같은 일이 발생해야 함을 정의합니다. `TridentBackendConfig` 삭제되었습니다. 다음 두 가지 값 중 하나를 취할 수 있습니다.
  - `delete`: 이로 인해 두 가지 모두 삭제됩니다. `TridentBackendConfig` CR 및 관련 백엔드. 이는 기본값입니다.
  - `retain`: 언제 `TridentBackendConfig` CR이 삭제되면 백엔드 정의는 여전히 존재하며 다음을 통해

관리할 수 있습니다. `tridentctl` . 삭제 정책을 다음으로 설정 `retain` 사용자가 이전 릴리스(21.04 이전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있도록 합니다. 이 필드의 값은 다음에 업데이트될 수 있습니다. `TridentBackendConfig` 생성됩니다.

백엔드의 이름은 다음을 사용하여 설정됩니다. `spec.backendName` . 지정하지 않으면 백엔드 이름이 다음 이름으로 설정됩니다. `TridentBackendConfig` 객체(메타데이터.이름). 백엔드 이름을 명시적으로 설정하는 것이 좋습니다. `spec.backendName` .

로 생성된 백엔드 `tridentctl` 연관된 것이 없습니다 `TridentBackendConfig` 물체. 이러한 백엔드를 관리하도록 선택할 수 있습니다. `kubectl` 생성하여 `TridentBackendConfig` 크.알. 동일한 구성 매개변수(예:)를 지정하는 데 주의해야 합니다. `spec.backendName` , `spec.storagePrefix` , `spec.storageDriverName` , 등). Trident 새로 생성된 항목을 자동으로 바인딩합니다. `TridentBackendConfig` 기존 백엔드를 사용합니다.

단계 개요

다음을 사용하여 새 백엔드를 생성하려면 `kubectl` , 다음을 수행해야 합니다.

1. 생성하다 "쿠버네티스 시크릿" . 비밀에는 Trident 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. 생성하다 `TridentBackendConfig` 물체. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 생성된 비밀을 참조합니다.

백엔드를 생성한 후에는 다음을 사용하여 상태를 관찰할 수 있습니다. `kubectl get tbc <tbc-name> -n <trident-namespace>` 그리고 추가적인 세부 정보를 수집합니다.

1단계: **Kubernetes Secret** 만들기

백엔드에 대한 액세스 자격 증명을 포함하는 비밀을 만듭니다. 이는 각 저장 서비스/플랫폼마다 고유합니다. 예를 들면 다음과 같습니다.

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

이 표는 각 저장 플랫폼의 비밀에 포함되어야 하는 필드를 요약한 것입니다.

저장 플랫폼 **Secret Fields** 설명	비밀	필드 설명
Azure NetApp Files	클라이언트ID	앱 등록의 클라이언트 ID
GCP용 Cloud Volumes Service	개인키_아이디	개인 키의 ID입니다. CVS 관리자 역할이 있는 GCP 서비스 계정의 API 키 일부
GCP용 Cloud Volumes Service	개인 키	개인 키. CVS 관리자 역할이 있는 GCP 서비스 계정의 API 키 일부
요소(NetApp HCI/ SolidFire)	엔드포인트	테넌트 자격 증명을 사용한 SolidFire 클러스터용 MVIP
ONTAP	사용자 이름	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됨
ONTAP	비밀번호	클러스터/SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됨
ONTAP	클라이언트 개인 키	클라이언트 개인 키의 Base64 인코딩된 값입니다. 인증서 기반 인증에 사용됨
ONTAP	chapUsername	수신 사용자 이름. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 개시자 비밀. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapTargetUsername	대상 사용자 이름. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 대상 개시자 비밀. useCHAP=true인 경우 필수입니다. 을 위한 `ontap-san` 그리고 `ontap-san-economy`

이 단계에서 생성된 비밀은 다음에 참조됩니다. `spec.credentials` 의 분야 `TridentBackendConfig` 다음 단계에서 생성되는 객체입니다.

2단계: 만들기 `TridentBackendConfig` 씨.씨.

이제 다음을 생성할 준비가 되었습니다. `TridentBackendConfig` 크.알. 이 예에서는 다음을 사용하는 백엔드 `ontap-san` 드라이버는 다음을 사용하여 생성됩니다. `TridentBackendConfig` 아래에 표시된 객체:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

3단계: 상태 확인 `TridentBackendConfig` 씨.씨.

이제 당신이 생성했습니다 `TridentBackendConfig` CR, 상태를 확인할 수 있습니다. 다음 예를 참조하세요.

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                      BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

백엔드가 성공적으로 생성되어 바인딩되었습니다. `TridentBackendConfig` 크.알.

위상은 다음 값 중 하나를 취할 수 있습니다.

- `Bound`: 그 `TridentBackendConfig` CR은 백엔드와 연결되어 있으며 해당 백엔드에는 다음이 포함됩니다. `configRef` 로 설정 `TridentBackendConfig` CR의 uid.
- `Unbound`: 를 사용하여 표현됨 `""` . 그만큼 `TridentBackendConfig` 객체가 백엔드에 바인딩되지 않았습니다. 모두 새로 생성됨 `TridentBackendConfig` CR은 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 Unbound로 돌아갈 수 없습니다.
- `Deleting`: 그 `TridentBackendConfig` CR의 `deletionPolicy` 삭제되도록 설정되었습니다. 때 `TridentBackendConfig` CR이 삭제되면 삭제 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVC)이 없는 경우 삭제 `TridentBackendConfig` Trident 백엔드와 다음을

삭제하게 됩니다. `TridentBackendConfig` 크.알.

  - 백엔드에 하나 이상의 PVC가 있는 경우 삭제 상태로 전환됩니다. 그만큼 `TridentBackendConfig` CR도 이후 삭제 단계로 들어갑니다. 백엔드와 `TridentBackendConfig` 모든 PVC가 삭제된 후에만 삭제됩니다.

- `Lost`: 연관된 백엔드 `TridentBackendConfig` CR이 실수로 또는 의도적으로 삭제되었으며 `TridentBackendConfig` CR에는 삭제된 백엔드에 대한 참조가 여전히 있습니다. 그만큼 `TridentBackendConfig` CR은 여전히 삭제될 수 있습니다. `deletionPolicy` 값.
- `Unknown`: Trident 백엔드의 상태 또는 존재를 확인할 수 없습니다. `TridentBackendConfig` 크.알. 예를 들어, API 서버가 응답하지 않거나 `tridentbackends.trident.netapp.io` CRD가 없습니다. 여기에는 개입이 필요할 수 있습니다.

이 단계에서는 백엔드가 성공적으로 생성되었습니다! 추가로 처리할 수 있는 작업은 다음과 같습니다."백엔드 업데이트 및 백엔드 삭제" .

(선택 사항) 4단계: 자세한 내용 보기

다음 명령을 실행하면 백엔드에 대한 자세한 정보를 얻을 수 있습니다.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME         BACKEND UUID
PHASE    STATUS     STORAGE DRIVER    DELETION POLICY
backend-tbc-ontap-san    ontap-san-backend    8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8    Bound    Success    ontap-san         delete
```

또한 YAML/JSON 덤프도 얻을 수 있습니다. `TridentBackendConfig` .

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo`포함하다 `backendName` 그리고 `backendUUID` 응답으로 생성된 백엔드의 `TridentBackendConfig` 크.알. 그만큼 `lastOperationStatus` 필드는 마지막 작업의 상태를 나타냅니다. `TridentBackendConfig` 사용자가 트리거할 수 있는 CR(예: 사용자가 무언가를 변경한 경우) `spec`) 또는 Trident에 의해 트리거됩니다(예: Trident 재시작 중). 성공이 될 수도 있고 실패가 될 수도 있습니다. `phase` 관계의 상태를 나타냅니다. `TridentBackendConfig` CR과 백엔드. 위의 예에서, `phase` Bound 값을 가지는데, 이는 다음을 의미합니다. `TridentBackendConfig` CR은 백엔드와 연관되어 있습니다.

당신은 실행할 수 있습니다 `kubectl -n trident describe tbc <tbc-cr-name>` 이벤트 로그의 세부 정보를 얻는 명령입니다.

> 연관된 백엔드를 포함하는 백엔드를 업데이트하거나 삭제할 수 없습니다. `TridentBackendConfig` 객체를 사용하여 `tridentctl` . 전환에 관련된 단계를 이해하려면 `tridentctl` 그리고 `TridentBackendConfig` ,"여기를 보세요" .

백엔드 관리

다음을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보세요. `kubectl` .

백엔드 삭제

삭제하여 `TridentBackendConfig` , Trident 백엔드를 삭제/보관하도록 지시합니다(기반 `deletionPolicy`). 백엔드를 삭제하려면 다음을 확인하세요. `deletionPolicy` 삭제되도록 설정되어 있습니다. 삭제하려면 `TridentBackendConfig` , 확인하십시오 `deletionPolicy` 유지되도록 설정되어 있습니다. 이렇게 하면 백엔드가 여전히 존재하고 다음을 사용하여 관리할 수 있습니다. `tridentctl` .

다음 명령을 실행하세요.

```
kubectl delete tbc <tbc-name> -n trident
```

Trident 사용 중이던 Kubernetes Secret을 삭제하지 않습니다. `TridentBackendConfig` . Kubernetes 사용자는 비밀을 정리할 책임이 있습니다. 비밀을 삭제할 때는 주의해야 합니다. 백엔드에서 사용하지 않는 비밀만 삭제해야 합니다.

기존 백엔드 보기

다음 명령을 실행하세요.

```
kubectl get tbc -n trident
```

또한 실행할 수도 있습니다 `tridentctl get backend -n trident` 또는 `tridentctl get backend -o yaml -n trident` 존재하는 모든 백엔드 목록을 얻습니다. 이 목록에는 또한 생성된 백엔드가 포함됩니다. `tridentctl` .

백엔드 업데이트

백엔드를 업데이트하는 데에는 여러 가지 이유가 있을 수 있습니다.

- 저장 시스템의 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 Kubernetes Secret이 사용됩니다. `TridentBackendConfig` 객체를 업데이트해야 합니다. Trident 제공된 최신 자격 증명으로 백엔드를 자동으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret을 업데이트합니다.

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- 매개변수(사용 중인 ONTAP SVM의 이름 등)를 업데이트해야 합니다.
  - 업데이트할 수 있습니다 `TridentBackendConfig` 다음 명령을 사용하여 Kubernetes를 통해 직접 객체를 가져올 수 있습니다.

    ```
    kubectl apply -f <updated-backend-file.yaml>
    ```

◦ 또는 기존 내용을 변경할 수 있습니다. `TridentBackendConfig` 다음 명령을 사용하여 CR을 실행합니다.

```
kubectl edit tbc <tbc-name> -n trident
```

- 백엔드 업데이트가 실패하면 백엔드는 마지막으로 알려진 구성을 그대로 유지합니다. 원인을 확인하려면 다음을 실행하여 로그를 볼 수 있습니다. `kubectl get tbc <tbc-name> -o yaml -n trident` 또는 `kubectl describe tbc <tbc-name> -n trident` .
- 구성 파일의 문제를 파악하고 수정한 후 업데이트 명령을 다시 실행할 수 있습니다.

tridentctl로 백엔드 관리 수행

다음을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보세요. `tridentctl` .

백엔드 만들기

생성한 후"백엔드 구성 파일" 다음 명령을 실행합니다.

```
tridentctl create backend -f <backend-file> -n trident
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후에는 간단히 다음을 실행할 수 있습니다. `create` 다시 명령을 내리세요.

백엔드 삭제

Trident 에서 백엔드를 삭제하려면 다음을 수행하세요.

1. 백엔드 이름을 검색합니다.

```
tridentctl get backend -n trident
```

2. 백엔드를 삭제합니다.

```
tridentctl delete backend <backend-name> -n trident
```

Trident 이 백엔드에서 여전히 존재하는 볼륨과 스냅샷을 프로비저닝한 경우 백엔드를 삭제하면 새 볼륨을 프로비저닝할 수 없습니다. 백엔드는 "삭제 중" 상태로 계속 존재합니다.

기존 백엔드 보기

Trident 알고 있는 백엔드를 보려면 다음을 수행하세요.

- 요약을 보려면 다음 명령을 실행하세요.

```
tridentctl get backend -n trident
```

- 모든 세부 정보를 보려면 다음 명령을 실행하세요.

```
tridentctl get backend -o json -n trident
```

백엔드 업데이트

새로운 백엔드 구성 파일을 만든 후 다음 명령을 실행합니다.

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

백엔드 업데이트가 실패한 경우 백엔드 구성에 문제가 있거나 잘못된 업데이트를 시도한 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후에는 간단히 다음을 실행할 수 있습니다. `update` 다시 명령을 내리세요.

백엔드를 사용하는 스토리지 클래스를 식별합니다.

이는 JSON으로 답할 수 있는 질문 종류의 예입니다. `tridentctl` 백엔드 객체에 대한 출력. 이것은 다음을 사용합니다 `jq` 설치해야 하는 유틸리티입니다.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

이는 다음을 사용하여 생성된 백엔드에도 적용됩니다. `TridentBackendConfig` .

백엔드 관리 옵션 간 이동

Trident 에서 백엔드를 관리하는 다양한 방법에 대해 알아보세요.

백엔드 관리를 위한 옵션

의 도입으로 `TridentBackendConfig` 이제 관리자는 백엔드를 관리하는 두 가지 고유한 방법을 사용할 수 있습니다. 이는 다음과 같은 질문을 제기합니다.

- 백엔드를 사용하여 생성할 수 있습니까? `tridentctl` ~로 관리되다 `TridentBackendConfig` ?
- 백엔드를 사용하여 생성할 수 있습니까? `TridentBackendConfig` 사용하여 관리됩니다 `tridentctl` ?

관리하다 `tridentctl` 백엔드를 사용하여 `TridentBackendConfig`

이 섹션에서는 다음을 사용하여 생성된 백엔드를 관리하는 데 필요한 단계를 다룹니다. `tridentctl` Kubernetes 인터페이스를 통해 직접 생성 `TridentBackendConfig` 사물.

이는 다음 시나리오에 적용됩니다.

- 기존 백엔드가 없는 경우 `TridentBackendConfig` 왜냐하면 그들은 ~로 창조되었기 때문이다 `tridentctl` .
- 새로운 백엔드가 생성되었습니다. `tridentctl` , 다른 `TridentBackendConfig` 객체가 존재합니다.

두 시나리오 모두 백엔드는 계속 존재하며 Trident 볼륨을 예약하고 이를 운영합니다. 관리자는 다음 두 가지 선택 중 하나를 선택할 수 있습니다.

- 계속 사용 `tridentctl` 이를 사용하여 생성된 백엔드를 관리합니다.
- 다음을 사용하여 생성된 백엔드 바인딩 `tridentctl` 새로운 것에 `TridentBackendConfig` 물체. 그렇게 하면 백엔드가 다음을 사용하여 관리됩니다. `kubectl` 그리고 아니다 `tridentctl` .

기존 백엔드를 관리하려면 다음을 사용합니다. `kubectl` , 당신은 만들어야 할 것입니다 `TridentBackendConfig` 기존 백엔드에 바인딩됩니다. 작동 원리는 다음과 같습니다.

1. Kubernetes Secret을 생성합니다. 비밀에는 Trident 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. 생성하다 `TridentBackendConfig` 물체. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 생성된 비밀을 참조합니다. 동일한 구성 매개변수(예:)를 지정하는 데 주의해야 합니다. `spec.backendName` , `spec.storagePrefix` , `spec.storageDriverName` , 등). `spec.backendName` 기존 백엔드의 이름으로 설정해야 합니다.

0단계**:** 백엔드 식별

생성하려면 `TridentBackendConfig` 기존 백엔드에 바인딩하려면 백엔드 구성을 얻어야 합니다. 이 예에서는 다음 JSON 정의를 사용하여 백엔드가 생성되었다고 가정해 보겠습니다.

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                  UUID
| STATE   | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

1단계: **Kubernetes Secret** 만들기

다음 예와 같이 백엔드에 대한 자격 증명을 포함하는 비밀을 만듭니다.

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

2단계: 만들기 `TridentBackendConfig` 씨.씨.

다음 단계는 다음을 만드는 것입니다. `TridentBackendConfig` 기존에 자동으로 바인딩되는 CR `ontap-nas-backend` (이 예에서처럼). 다음 요구 사항이 충족되는지 확인하세요.

- 동일한 백엔드 이름이 정의되어 있습니다. `spec.backendName` .
- 구성 매개변수는 원래 백엔드와 동일합니다.
- 가상 풀(존재하는 경우)은 원래 백엔드와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우에는 `TridentBackendConfig` 다음과 같이 보일 것입니다:

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

3단계: 상태 확인 `TridentBackendConfig` 씨.씨.

후에 `TridentBackendConfig` 생성되었으므로 해당 단계는 다음과 같아야 합니다. `Bound` . 또한 기존 백엔드와 동일한 백엔드 이름과 UUID를 반영해야 합니다.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

백엔드는 이제 다음을 사용하여 완전히 관리됩니다. `tbc-ontap-nas-backend` `TridentBackendConfig` 물체.

관리하다 `TridentBackendConfig` 백엔드를 사용하여 `tridentctl`

`tridentctl`를 사용하여 생성된 백엔드를 나열하는 데 사용할 수 있습니다. `TridentBackendConfig` . 또한 관리자는 다음을 통해 이러한 백엔드를 완전히 관리하도록 선택할 수도 있습니다. `tridentctl` 삭제하여 `TridentBackendConfig` 그리고 확인하다 `spec.deletionPolicy` 로 설정됩니다 `retain` .

0단계: 백엔드 식별

예를 들어, 다음 백엔드가 다음을 사용하여 생성되었다고 가정해 보겠습니다. `TridentBackendConfig` :

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

출력에서 다음이 표시됩니다. `TridentBackendConfig` 성공적으로 생성되었으며 백엔드에 바인딩되었습니다. [백엔드의 UUID를 확인하세요]

1단계: 확인 `deletionPolicy` 로 설정됩니다 `retain`

의 가치를 살펴보자 `deletionPolicy` . 이것은 설정되어야 합니다 `retain` . 이것은 다음을 보장합니다. `TridentBackendConfig` CR이 삭제되면 백엔드 정의는 여전히 존재하며 다음을 통해 관리할 수 있습니다. `tridentctl` .

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

다음 단계로 진행하지 마십시오. `deletionPolicy` 로 설정됩니다 `retain` .

2단계: 삭제 `TridentBackendConfig` 씨.씨.

마지막 단계는 삭제하는 것입니다. `TridentBackendConfig` 크.알. 확인 후 `deletionPolicy` 로 설정됩니다 `retain` , 삭제를 진행할 수 있습니다.

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

삭제 시 `TridentBackendConfig` 객체를 제거하면 Trident 백엔드 자체를 삭제하지 않고 객체를 제거합니다.

스토리지 클래스 생성 및 관리

스토리지 클래스 생성

Kubernetes StorageClass 객체를 구성하고 Trident 가 볼륨을 프로비저닝하는 방법을 지시하는 스토리지 클래스를 만듭니다.

Kubernetes StorageClass 객체 구성

그만큼 "Kubernetes StorageClass 객체" Trident 해당 클래스에 사용되는 프로비저너로 식별하고 Trident 볼륨을 프로비저닝하는 방법을 지시합니다. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

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

스토리지 클래스 생성

StorageClass 객체를 만든 후에는 스토리지 클래스를 만들 수 있습니다. 저장 클래스 샘플 사용하거나 수정할 수 있는 몇 가지 기본 샘플을 제공합니다.

단계

1. 이것은 Kubernetes 객체이므로 다음을 사용합니다. `kubectl` Kubernetes에서 생성하세요.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스를 볼 수 있어야 하며, Trident 가 백엔드에서 풀을 검색했어야 합니다.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

저장 클래스 샘플

Trident 제공합니다 "특정 백엔드에 대한 간단한 스토리지 클래스 정의" .

또는 편집할 수 있습니다 `sample-input/storage-class-csi.yaml.templ` 설치 프로그램과 함께 제공되는 파일을 교체합니다. *`BACKEND_TYPE`* 저장소 드라이버 이름으로.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

스토리지 클래스 관리

기존 스토리지 클래스를 보고, 기본 스토리지 클래스를 설정하고, 스토리지 클래스 백엔드를 식별하고, 스토리지 클래스를 삭제할 수 있습니다.

기존 스토리지 클래스 보기

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
kubectl get storageclass <storage-class> -o json
```

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하세요.

```
tridentctl get storageclass <storage-class> -o json
```

기본 저장소 클래스 설정

Kubernetes 1.6에서는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 이는 사용자가 PVC(영구 볼륨 클레임)에서 영구 볼륨을 지정하지 않은 경우 영구 볼륨을 프로비저닝하는 데 사용되는 스토리지 클래스입니다.

- 주석을 설정하여 기본 저장 클래스를 정의합니다. `storageclass.kubernetes.io/is-default-class` 저장 클래스 정의에서 true로 설정합니다. 사양에 따르면, 다른 값이나 주석이 없는 경우 거짓으로 해석됩니다.
- 다음 명령을 사용하여 기존 스토리지 클래스를 기본 스토리지 클래스로 구성할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 마찬가지로 다음 명령을 사용하여 기본 저장소 클래스 주석을 제거할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 설치 프로그램 번들에는 이 주석이 포함된 예도 있습니다.

> 클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes는 기술적으로 두 개 이상을 갖는 것을 막지는 않지만, 기본 스토리지 클래스가 전혀 없는 것처럼 동작합니다.

스토리지 클래스의 백엔드 식별

이는 JSON으로 답할 수 있는 질문 종류의 예입니다. `tridentctl` Trident 백엔드 객체에 대한 출력. 이것은 다음을 사용합니다 `jq` 먼저 설치해야 할 유틸리티가 있습니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

스토리지 클래스 삭제

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행하세요.

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`저장 클래스로 대체해야 합니다.

이 스토리지 클래스를 통해 생성된 모든 영구 볼륨은 변경되지 않으며 Trident 계속해서 이를 관리합니다.

> Trident 공백을 강화합니다. `fsType` 그것이 만들어내는 볼륨 때문에요. iSCSI 백엔드의 경우 다음을 적용하는 것이 좋습니다. `parameters.fsType` StorageClass에서. 기존 StorageClass를 삭제하고 다시 생성해야 합니다. `parameters.fsType` 지정된.

볼륨 제공 및 관리

볼륨 제공

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

개요

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

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

PVC를 만든 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 만듭니다

단계

1. PVC을 생성합니다.

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

2. PVC 상태를 확인하세요.

```
kubectl get pvc
```

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

1. 볼륨을 포드에 마운트합니다.

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

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

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

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

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

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

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

다음 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWO 접근이 가능한 **PVC**

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

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

NVMe/TCP를 사용한 **PVC**

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

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

Pod 매니페스트 샘플

이러한 예는 PVC를 포드에 부착하는 기본 구성을 보여줍니다.

기본 구성

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

기본 **NVMe/TCP** 구성

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

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

볼륨 확장

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

iSCSI 볼륨 확장

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

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

1단계: 볼륨 확장을 지원하도록 **StorageClass** 구성

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

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

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

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

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

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

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

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

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

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

3단계: **PVC**를 부착하는 포드 정의

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

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

이 예에서는 다음을 사용하는 포드가 생성됩니다. `san-pvc` .

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

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

4단계: **PV** 확장

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

```
kubectl edit pvc san-pvc
```

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

5단계: 확장 검증

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

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

FC 볼륨 확장

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

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

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

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

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

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

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

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

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

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

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

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

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

kubectl get pv
NAME                                       CAPACITY    ACCESS MODES
RECLAIM POLICY    STATUS    CLAIM              STORAGECLASS    REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi         RWO
Delete            Bound     default/san-pvc    ontap-san                 10s
```

3단계: **PVC**를 부착하는 포드 정의

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

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

이 예에서는 다음을 사용하는 포드가 생성됩니다. `san-pvc` .

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

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

4단계: **PV** 확장

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

```
kubectl edit pvc san-pvc
```

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

5단계: 확장 검증

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

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

NFS 볼륨 확장

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

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

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

```
cat storageclass-ontapnas.yaml
```

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

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

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

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

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

Trident 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES    STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

3단계: **PV** 확장

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

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

4단계: 확장 검증

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

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

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

수입량

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

개요 및 고려 사항

볼륨을 Trident 로 가져와서 다음을 수행할 수 있습니다.

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

고려 사항

볼륨을 가져오기 전에 다음 고려 사항을 검토하세요.

- Trident RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 대상 볼륨입니다. 볼륨을 Trident 로 가져오기 전에 미러 관계를 해제해야 합니다.

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

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

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

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

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

볼륨 가져오기

사용할 수 있습니다 `tridentctl import` 볼륨을 가져오려면.

단계

1. 영구 볼륨 클레임(PVC) 파일을 만듭니다(예: `pvc.yaml`) PVC를 만드는 데 사용됩니다. PVC 파일에는 다음이 포함되어야 합니다. `name` , `namespace` , `accessModes` , 그리고 `storageClassName` . 선택적으로 지정할 수 있습니다. `unixPermissions` PVC 정의에서.

   다음은 최소 사양의 예입니다.

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

PV 이름이나 볼륨 크기와 같은 추가 매개변수를 포함하지 마세요. 이로 인해 가져오기 명령이 실패할 수 있습니다.

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

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

예시

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하세요.

ONTAP NAS 및 ONTAP NAS FlexGroup

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

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

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

ONTAP NAS 예시

다음은 관리되는 볼륨과 관리되지 않는 볼륨 가져오기의 예를 보여줍니다.

관리되는 볼륨

다음 예제에서는 이름이 지정된 볼륨을 가져옵니다. `managed_volume` 백엔드에 이름이 지정됨 `ontap_nas` :

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
```

관리되지 않는 볼륨

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

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

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
```

ONTAP 산

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

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

볼륨.

ONTAP SAN 예시

다음은 관리되는 볼륨과 관리되지 않는 볼륨 가져오기의 예를 보여줍니다.

관리되는 볼륨

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

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

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

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

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

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

다음 예와 같이 Kubernetes 노드 IQN과 IQN을 공유하는 igroup에 LUN이 매핑된 경우 오류가 발생합니다. `LUN`

`already mapped to initiator(s) in this group`. 볼륨을 가져오려면 초기자를 제거하거나 LUN의 매핑을 해제해야 합니다.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

요소

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

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

요소 예

다음 예제에서는 다음을 가져옵니다. `element-managed` 백엔드의 볼륨 `element_default` .

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

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

구글 클라우드 플랫폼

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

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

Google Cloud Platform 예시

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

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

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

Azure NetApp Files

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

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

Azure NetApp Files 예제

다음 예제에서는 다음을 가져옵니다. `azure-netapp-files` 백엔드의 볼륨 `azurenetappfiles_40517` 볼륨 경로로 `importvol1` .

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |              BACKEND UUID               | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `google-cloud-netapp-volumes` 운전사.

Google Cloud NetApp Volumes 예시

다음 예제에서는 다음을 가져옵니다. `google-cloud-netapp-volumes` 백엔드의 볼륨 `backend-tbc-gcnv1` 볼륨과 함께 `testvoleasiaeast1` .

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

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

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

볼륨 이름 및 레이블 사용자 정의

Trident 사용하면 생성한 볼륨에 의미 있는 이름과 레이블을 지정할 수 있습니다. 이를 통해 볼륨을 식별하고 해당 Kubernetes 리소스(PVC)에 쉽게 매핑할 수 있습니다. 사용자 정의 볼륨 이름과 사용자 정의 레이블을 생성하기 위해 백엔드 수준에서 템플릿을 정의할 수도 있습니다. 생성, 가져오기 또는 복제하는 모든 볼륨은 템플릿을 준수합니다.

시작하기 전에

사용자 정의 가능한 볼륨 이름 및 레이블 지원:

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

제한 사항

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

사용자 정의 가능한 볼륨 이름의 주요 동작

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

이름 템플릿 및 레이블을 사용한 백엔드 구성 예

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

루트 수준 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

풀 레벨 예시

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

이름 템플릿 예시

예시 **1**:

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

예시 **2**:

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

고려할 사항

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

네임스페이스 간에 **NFS** 볼륨 공유

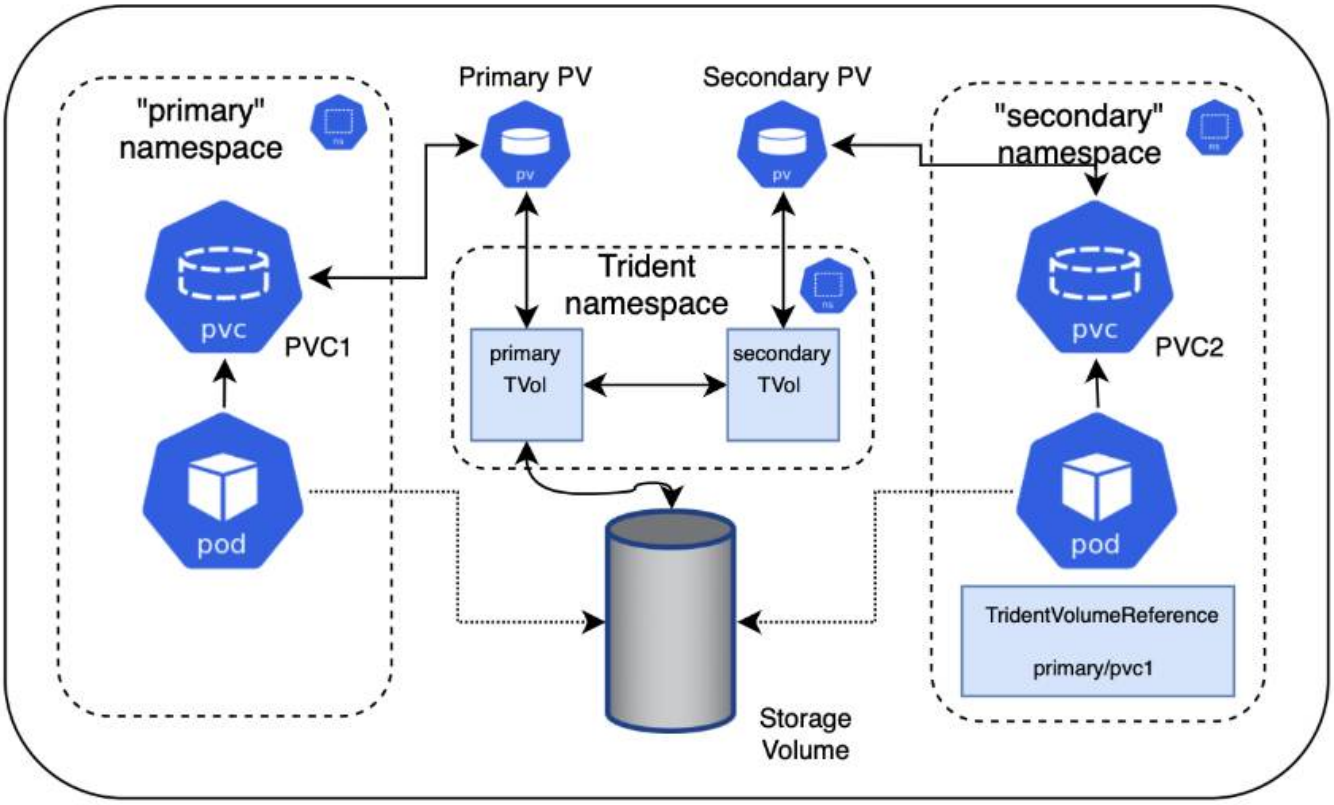
Trident 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

특징

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

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

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



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1 볼륨을 공유하도록 소스 **PVC**를 구성합니다.

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

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

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

3 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다.

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4 대상 네임스페이스에 하위 **PVC**를 만듭니다.

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

소스 및 대상 네임스페이스 구성

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

단계

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

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

Trident PV와 백엔드 NFS 스토리지 볼륨을 생성합니다.

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

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

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

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

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

> 대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

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

공유 볼륨 삭제

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

사용 `tridentctl get` 하위 볼륨을 쿼리하려면

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

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

플래그:

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

제한 사항

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

더 많은 정보를 원하시면

네임스페이스 간 볼륨 액세스에 대해 자세히 알아보려면 다음을 참조하세요.

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

네임스페이스 전체에서 볼륨 복제

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

필수 조건

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

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

빠른 시작

몇 단계만 거치면 볼륨 복제를 설정할 수 있습니다.

1. 볼륨을 복제하기 위해 소스 **PVC**를 구성합니다.

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

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

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

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

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4. 대상 네임스페이스에 복제 **PVC**를 만듭니다.

대상 네임스페이스의 소유자는 소스 네임스페이스에서 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스 구성

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

단계

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

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

Trident PV와 백엔드 스토리지 볼륨을 생성합니다.

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

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

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

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

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

제한 사항

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

SnapMirror 사용하여 볼륨 복제

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

- 볼륨(PVC) 간 미러 관계 생성
- 볼륨 간 미러 관계 제거
- 거울 속 관계를 깨세요
- 재해 상황(장애 조치) 동안 보조 볼륨을 홍보합니다.

- 계획된 장애 조치 또는 마이그레이션 중 클러스터 간에 애플리케이션의 손실 없는 전환을 수행합니다.

복제 전제 조건

시작하기 전에 다음 전제 조건이 충족되었는지 확인하세요.

ONTAP 클러스터

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

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

  SnapMirror 비동기 보호만 지원됩니다.

피어링

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

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

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

지원되는 드라이버

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

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

거울 **PVC** 만들기

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

단계

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

예

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

b. 이전에 생성한 StorageClass로 PVC를 생성합니다.

예

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

c. 지역 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

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

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

d. TridentMirrorRelationship CR을 가져와서 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

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

      예

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

   b. 목적지 및 소스 정보를 사용하여 MirrorRelationship CR을 만듭니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

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

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

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

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

볼륨 복제 상태

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

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

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

계획되지 않은 장애 조치 중 **2**차 **PVC** 홍보

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

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

계획된 장애 조치 중 보조 **PVC** 홍보

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

단계

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

   예

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

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

장애 조치 후 미러 관계 복원

미러 관계를 복원하기 전에 새로운 기본 관계로 만들려는 측면을 선택하세요.

단계

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

추가 작업

Trident 기본 및 보조 볼륨에서 다음 작업을 지원합니다.

1차 PVC를 새로운 2차 PVC로 복제합니다.

기본 PVC와 보조 PVC가 이미 있는지 확인하세요.

단계

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

미러링된 1차 또는 2차 PVC 크기 조정

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

PVC에서 복제 제거

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행합니다.

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

PVC(이전에 미러링됨) 삭제

Trident 복제된 PVC를 확인하고 볼륨을 삭제하기 전에 복제 관계를 해제합니다.

TMR 삭제

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

ONTAP 이 온라인일 때 미러 관계 업데이트

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

ONTAP 이 오프라인일 때 미러 관계 업데이트

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

예

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

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

CSI 토폴로지 사용

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

개요

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

CSI 토폴로지 기능에 대해 자세히 알아보세요 "여기" .

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

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

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

필요한 것

CSI 토폴로지를 활용하려면 다음이 필요합니다.

- Kubernetes 클러스터를 실행 중"지원되는 Kubernetes 버전"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

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

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

1단계: 토폴로지 인식 백엔드 만들기

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

다음은 백엔드 정의의 예입니다.

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

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

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

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

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

2단계: 토폴로지를 인식하는 **StorageClass** 정의

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

다음 예를 참조하세요.

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

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

3단계: **PVC** 만들기 및 사용

StorageClass를 생성하고 백엔드에 매핑했으므로 이제 PVC를 생성할 수 있습니다.

예를 보세요 `spec` 아래에:

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

이 매니페스트를 사용하여 PVC를 생성하면 다음과 같은 결과가 발생합니다.

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

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

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

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

다음 출력을 확인하세요.

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

백엔드를 업데이트하여 포함합니다. `supportedTopologies`

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

더 많은 정보를 찾아보세요

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

스냅샷 작업

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

개요

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

시작하기 전에

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

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

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

볼륨 스냅샷 생성

단계

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

     예

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

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

2. 기존 PVC의 스냅샷을 만듭니다.

   예시
   - 이 예제에서는 기존 PVC의 스냅샷을 만듭니다.

     ```
     cat snap.yaml
     ```

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

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

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

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

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

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

볼륨 스냅샷에서 **PVC** 생성

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

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

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

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

볼륨 스냅샷 가져오기

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

시작하기 전에

Trident 스냅샷의 부모 볼륨을 생성하거나 가져와야 합니다.

단계

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

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

     예

     다음 예제에서는 다음을 생성합니다. `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 객체 `snap-01` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. 클러스터 관리자: 생성 `VolumeSnapshot` 참조하는 CR `VolumeSnapshotContent` 물체. 이것은 사용에 대한 액세스를 요청합니다. `VolumeSnapshot` 주어진 네임스페이스에서.

   예

   다음 예제에서는 다음을 생성합니다. `VolumeSnapshot` CR이 명명됨 `import-snap` 참조하는 `VolumeSnapshotContent` 명명된 `import-snap-content` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

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

예

다음 예제에서는 다음을 참조하는 PVC를 생성합니다. `VolumeSnapshot` 명명된 `import-snap` .

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

스냅샷을 사용하여 볼륨 데이터 복구

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

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

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

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

스냅샷에서 제자리 볼륨 복원

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

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

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인하세요.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인하세요.

```
kubectl get vs
```

단계

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

   TASR CR은 PVC 및 VS가 있는 네임스페이스에 있어야 합니다.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

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

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

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

결과

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

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

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

연관된 스냅샷이 있는 **PV** 삭제

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

볼륨 스냅샷 컨트롤러 배포

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

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

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

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

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

관련 링크

- "볼륨 스냅샷"
- "볼륨 스냅샷 클래스"

볼륨 그룹 스냅샷 작업

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

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

볼륨 그룹 스냅샷 생성

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

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

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

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

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

단계

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

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

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

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

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

  이 예제에서는 볼륨 그룹 스냅샷을 생성합니다.

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

그룹 스냅샷을 사용하여 볼륨 데이터 복구

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

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

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

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

스냅샷에서 제자리 볼륨 복원

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

자세한 내용은 다음을 참조하세요. "스냅샷에서 제자리 볼륨 복원".

연관된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

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

볼륨 스냅샷 컨트롤러 배포

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

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

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

관련 링크

- "볼륨 그룹 스냅샷 클래스"
- "볼륨 스냅샷"

Trident 관리 및 모니터링

Trident 업그레이드

Trident 업그레이드

Trident 24.02 릴리스부터 4개월 릴리스 주기를 따르며, 매년 3개의 주요 릴리스를 제공합니다. 각각의 새로운 릴리스는 이전 릴리스를 기반으로 하며 새로운 기능, 성능 향상, 버그 수정 및 개선 사항을 제공합니다. Trident 의 새로운 기능을 활용하려면 최소한 1년에 한 번 업그레이드하는 것이 좋습니다.

업그레이드 전 고려 사항

Trident 의 최신 릴리스로 업그레이드할 때 다음 사항을 고려하세요.

- 주어진 Kubernetes 클러스터의 모든 네임스페이스에 Trident 인스턴스는 하나만 설치되어야 합니다.
- Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 더 이상 알파 또는 베타 스냅샷을 지원하지 않습니다.
- Google Cloud에 대한 Cloud Volumes Service 생성한 경우"CVS 서비스 유형" , 백엔드 구성을 업데이트하여 사용해야 합니다. `standardsw` 또는 `zoneredundantstandardsw` Trident 23.01에서 업그레이드할 때의 서비스 수준. 업데이트 실패 `serviceLevel` 백엔드에서는 볼륨이 실패할 수 있습니다. 참조하다 "CVS 서비스 유형 샘플" 자세한 내용은.
- 업그레이드할 때 다음을 제공하는 것이 중요합니다. `parameter.fsType` ~에 `StorageClasses` Trident 에서 사용됨. 삭제하고 다시 만들 수 있습니다 `StorageClasses` 기존 볼륨을 방해하지 않고.
  - 이는 시행을 위한 요구사항입니다. "보안 컨텍스트" SAN 볼륨의 경우.
  - https://github.com/NetApp/trident/tree/master/trident-installer/sample-input [sample input^] 디렉토리에는 https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templ 과 같은 예가 포함되어 있습니다.[`storage-class-basic.yaml.templ` ^] 및 링크: `storage-class-bronze-default.yaml` .
  - 자세한 내용은 다음을 참조하세요."알려진 문제" .

1단계**:** 버전 선택

Trident 버전은 날짜 기반을 따릅니다. `YY.MM` 명명 규칙에 따르면 "YY"는 연도의 마지막 두 자리이고 "MM"은 월입니다. Dot 릴리스는 다음을 따릅니다. `YY.MM.X` 여기서 "X"는 패치 수준입니다. 업그레이드할 버전은 업그레이드하려는 버전에 따라 선택됩니다.

- 설치된 버전의 4개 릴리스 창 내에 있는 모든 대상 릴리스로 직접 업그레이드를 수행할 수 있습니다. 예를 들어, 24.06(또는 모든 24.06 dot 릴리스)에서 25.06으로 직접 업그레이드할 수 있습니다.
- 4개 릴리스 창 이외의 릴리스에서 업그레이드하는 경우 여러 단계로 업그레이드하세요. 업그레이드 지침을 사용하세요."이전 버전" 4개 릴리스 창에 맞는 최신 릴리스로 업그레이드합니다. 예를 들어, 23.07을 실행 중이고 25.06으로 업그레이드하려는 경우:
  a. 23.07에서 24.06으로 첫 번째 업그레이드.
  b. 그런 다음 24.06에서 25.06으로 업그레이드하세요.

OpenShift Container Platform에서 Trident 연산자를 사용하여 업그레이드하는 경우 Trident 21.01.1 이상으로 업그레이드해야 합니다. 21.01.0과 함께 출시된 Trident 연산자에는 21.01.1에서 수정된 알려진 문제가 포함되어 있습니다. 자세한 내용은 다음을 참조하세요. "GitHub의 이슈 세부 정보" .

2단계**:** 원래 설치 방법 확인

Trident 처음 설치하는 데 사용한 버전을 확인하려면 다음을 수행하세요.

1. 사용 `kubectl get pods -n trident` 꼬투리를 조사하기 위해서.
   - 운영자 포드가 없는 경우 Trident 설치되었습니다. `tridentctl` .
   - 운영자 포드가 있는 경우, Trident 운영자를 수동으로 사용하거나 Helm을 사용하여 Trident 설치했습니다.
2. 운영자 포드가 있는 경우 사용하세요 `kubectl describe torc` Helm을 사용하여 Trident 설치되었는지 확인합니다.
   - Helm 레이블이 있으면 Helm을 사용하여 Trident 설치되었다는 의미입니다.
   - Helm 레이블이 없으면 Trident 연산자를 사용하여 Trident 수동으로 설치되었습니다.

3단계**:** 업그레이드 방법 선택

일반적으로 초기 설치에 사용한 것과 동일한 방법을 사용하여 업그레이드해야 하지만"설치 방법 간 이동" . Trident 업그레이드하는 데는 두 가지 옵션이 있습니다.

- "Trident 연산자를 사용하여 업그레이드"

검토해 보시기 바랍니다"운영자 업그레이드 워크플로 이해" 운영자와 함께 업그레이드하기 전에.

*

운영자와 함께 업그레이드하세요

운영자 업그레이드 워크플로 이해

Trident 운영자를 사용하여 Trident 업그레이드하기 전에 업그레이드 중에 발생하는 백그라운드 프로세스를 이해해야 합니다. 여기에는 롤링 업데이트를 활성화하는 Trident 컨트롤러, 컨트롤러 Pod 및 노드 Pod, 노드 DaemonSet에 대한 변경 사항이 포함됩니다.

Trident 운영자 업그레이드 처리

많은 것 중 하나"Trident 연산자 사용의 이점" Trident 설치하고 업그레이드하면 기존에 마운트된 볼륨을 방해하지 않고 Trident 와 Kubernetes 객체를 자동으로 처리할 수 있습니다. 이러한 방식으로 Trident 다운타임 없이 업그레이드를 지원할 수 있습니다."롤링 업데이트" . 특히, Trident 운영자는 Kubernetes 클러스터와 통신하여 다음을 수행합니다.

- Trident Controller 배포 및 노드 DaemonSet을 삭제하고 다시 생성합니다.
- Trident Controller Pod와 Trident Node Pod를 새 버전으로 교체하세요.
  - 노드가 업데이트되지 않더라도 나머지 노드가 업데이트되는 것은 방해가 되지 않습니다.
  - Trident Node Pod가 실행 중인 노드만 볼륨을 마운트할 수 있습니다.

Kubernetes 클러스터의 Trident 아키텍처에 대한 자세한 내용은 다음을 참조하세요."Trident 아키텍처" .

운영자 업그레이드 워크플로

Trident 연산자를 사용하여 업그레이드를 시작하는 경우:

1. * Trident 연산자*:
   a. 현재 설치된 Trident 버전(버전 *n*)을 감지합니다.
   b. CRD, RBAC, Trident SVC를 포함한 모든 Kubernetes 객체를 업데이트합니다.
   c. 버전 _n_에 대한 Trident Controller 배포를 삭제합니다.
   d. 버전 _n+1_에 대한 Trident Controller 배포를 생성합니다.
2. *Kubernetes*는 _n+1_에 대한 Trident Controller Pod를 생성합니다.
3. * Trident 연산자*:
   a. _n_에 대한 Trident Node DaemonSet을 삭제합니다. 운영자는 Node Pod 종료를 기다리지 않습니다.
   b. _n+1_에 대한 Trident Node Daemonset을 생성합니다.
4. *Kubernetes*는 Trident Node Pod _n_을 실행하지 않는 노드에 Trident Node Pod를 생성합니다. 이렇게 하면 어떤 버전의 Trident Node Pod가 노드에 두 개 이상 존재하지 않게 됩니다.

Trident Operator 또는 **Helm**을 사용하여 **Trident** 설치 업그레이드

Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 Trident 업그레이드할 수 있습니다. Trident 운영자 설치에서 다른 Trident 운영자 설치로 업그레이드하거나 다음에서 업그레이드할 수 있습니다. `tridentctl` Trident 운영자 버전에 설치. 검토"업그레이드 방법을 선택하세요" Trident 운영자 설치를 업그레이드하기 전에.

수동 설치 업그레이드

클러스터 범위 Trident Operator 설치에서 다른 클러스터 범위 Trident Operator 설치로 업그레이드할 수 있습니다. 모든 Trident 버전은 클러스터 범위 연산자를 사용합니다.

네임스페이스 범위 연산자(버전 20.07~20.10)를 사용하여 설치된 Trident 에서 업그레이드하려면 다음 업그레이드 지침을 사용하세요."설치된 버전" Trident 의.

이 작업에 관하여

Trident Kubernetes 버전에 대한 연산자를 설치하고 관련 객체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.

- Kubernetes 1.24를 실행하는 클러스터의 경우 다음을 사용하세요."bundle_pre_1_25.yaml" .
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 다음을 사용하세요."bundle_post_1_25.yaml" .

시작하기 전에

Kubernetes 클러스터를 실행 중인지 확인하세요."지원되는 Kubernetes 버전" .

단계

1. Trident 버전을 확인하세요:

```
./tridentctl -n trident version
```

2. 업데이트 `operator.yaml` , `tridentorchestrator_cr.yaml` , 그리고 `post_1_25_bundle.yaml` 업그레이드하려는 버전(예: 25.06)에 대한 레지스트리와 이미지 경로, 그리고 올바른 비밀번호를 사용합니다.
3. 현재 Trident 인스턴스를 설치하는 데 사용된 Trident 연산자를 삭제합니다. 예를 들어, 25.02에서 업그레이드하는 경우 다음 명령을 실행합니다.

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 초기 설치를 사용자 정의한 경우 `TridentOrchestrator` 속성을 편집할 수 있습니다 `TridentOrchestrator` 설치 매개변수를 수정하는 객체입니다. 여기에는 오프라인 모드에 대한 미러링된 Trident 및 CSI 이미지 레지스트리를 지정하고, 디버그 로그를 활성화하거나, 이미지 풀 비밀을 지정하기 위해 변경한 내용이 포함될 수 있습니다.
5. _<bundle.yaml>_이 있는 환경에 맞는 올바른 번들 YAML 파일을 사용하여 Trident 설치하세요. `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전에 따라 다릅니다. 예를 들어, Trident 25.06.0을 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. 트라이던트 토크를 편집하여 이미지 25.06.0을 포함시킵니다.

Helm 설치 업그레이드

Trident 헬름 설치를 업그레이드할 수 있습니다.

> Trident 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드하는 경우 values.yaml을 업데이트하여 설정해야 합니다. `excludePodSecurityPolicy` 에게 `true` 또는 추가 `--set excludePodSecurityPolicy=true` 에게 `helm upgrade` 클러스터를 업그레이드하기 전에 명령을 실행하세요.

Trident helm을 업그레이드하지 않고 이미 Kubernetes 클러스터를 1.24에서 1.25로 업그레이드한 경우 helm 업그레이드가 실패합니다. 헬름 업그레이드를 진행하려면 다음 단계를 필수 조건으로 수행하세요.

1. helm-mapkubeapis 플러그인을 설치하세요 https://github.com/helm/helm-mapkubeapis .
2. Trident 설치된 네임스페이스에서 Trident 릴리스에 대한 테스트 실행을 수행합니다. 여기에는 정리될 리소스가 나열됩니다.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. 헬름을 타고 전체 주행을 한 후 청소를 실시합니다.

```
helm mapkubeapis trident  --namespace trident
```

단계

1. 만약 당신이라면"Helm을 사용하여 Trident 설치했습니다." , 사용할 수 있습니다 `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0` 한 단계로 업그레이드하세요. Helm 저장소를 추가하지 않았거나 이를 사용하여 업그레이드할 수 없는 경우:
   a. 최신 Trident 릴리스를 다운로드하세요"GitHub의 *Assets* 섹션" .
   b. 사용하다 `helm upgrade` 명령은 어디에 `trident-operator-25.06.0.tgz` 업그레이드하려는 버전을 나타냅니다.

```
helm upgrade <name> trident-operator-25.06.0.tgz
```

> 초기 설치 중에 사용자 정의 옵션을 설정한 경우(예: Trident 및 CSI 이미지에 대한 개인 미러링 레지스트리 지정) 다음을 추가합니다. `helm upgrade` 명령을 사용하여 `--set` 해당 옵션이 업그레이드 명령에 포함되어 있는지 확인하세요. 그렇지 않으면 값이 기본값으로 재설정됩니다.

2. 달리다 `helm list` 차트와 앱 버전이 모두 업그레이드되었는지 확인하세요. 달리다 `tridentctl logs` 디버그 메시지를 검토합니다.

업그레이드 `tridentctl` Trident 운영자에 설치

Trident 운영자의 최신 릴리스로 업그레이드할 수 있습니다. `tridentctl` 설치. 기존 백엔드와 PVC는 자동으로 사용 가능합니다.

> 설치 방법을 전환하기 전에 다음을 검토하세요."설치 방법 간 이동" .

단계

1. 최신 Trident 릴리스를 다운로드하세요.

```
# Download the release required [25.06.0]
mkdir 25.06.0
cd 25.06.0
wget
https://github.com/NetApp/trident/releases/download/v25.06.0/trident-
installer-25.06.0.tar.gz
tar -xf trident-installer-25.06.0.tar.gz
cd trident-installer
```

2. 생성하다 `tridentorchestrator` 매니페스트의 CRD.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 동일한 네임스페이스에 클러스터 범위 연산자를 배포합니다.

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. 생성하다 `TridentOrchestrator` Trident 설치를 위한 CR.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Trident 의도한 버전으로 업그레이드되었는지 확인하세요.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.06.0
```

tridentctl로 업그레이드하세요

기존 Trident 설치를 쉽게 업그레이드할 수 있습니다. `tridentctl` .

이 작업에 관하여

Trident 제거하고 다시 설치하면 업그레이드가 수행됩니다. Trident 제거해도 Trident 배포에 사용되는 영구 볼륨 클레임(PVC) 및 영구 볼륨(PV)은 삭제되지 않습니다. 이미 프로비저닝된 PV는 Trident 가 오프라인인 동안에도 사용 가능하며, Trident 온라인 상태로 돌아온 후 그 사이에 생성된 모든 PVC에 대한 볼륨을 프로비저닝합니다.

시작하기 전에

검토"업그레이드 방법을 선택하세요" 업그레이드하기 전에 다음을 사용하세요. `tridentctl` .

단계

1. 제거 명령을 실행하세요 `tridentctl` CRD 및 관련 객체를 제외한 Trident 와 연관된 모든 리소스를 제거합니다.

```
./tridentctl uninstall -n <namespace>
```

2. Trident 다시 설치하세요. 참조하다"tridentctl을 사용하여 Trident 설치" .

업그레이드 과정을 중단하지 마세요. 설치 프로그램이 완전히 실행되었는지 확인하세요.

tridentctl을 사용하여 Trident 관리

그만큼 "Trident 설치 프로그램 번들" 포함 `tridentctl` Trident 에 간편하게 접근할 수 있는 명령줄 유틸리티입니다. 충분한 권한이 있는 Kubernetes 사용자는 이를 사용하여 Trident 설치하거나 Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

명령 및 글로벌 플래그

당신은 실행할 수 있습니다 `tridentctl help` 사용 가능한 명령 목록을 얻으려면 `tridentctl` 또는 추가 `--help` 특정 명령에 대한 옵션과 플래그 목록을 얻으려면 해당 명령에 플래그를 추가합니다.

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl` 유틸리티는 다음 명령과 글로벌 플래그를 지원합니다.

명령

create
Trident 에 리소스를 추가합니다.

delete
Trident 에서 하나 이상의 리소스를 제거합니다.

get
Trident 에서 하나 이상의 리소스를 받으세요.

help
모든 명령에 대한 도움말.

images
Trident 에 필요한 컨테이너 이미지 표를 인쇄하세요.

import
기존 리소스를 Trident 로 가져옵니다.

install
Trident 설치하세요.

logs
Trident 에서 로그를 인쇄합니다.

send
Trident 에서 리소스를 보내세요.

uninstall
Trident 제거합니다.

update
Trident 에서 리소스를 수정합니다.

update backend state
백엔드 작업을 일시적으로 중단합니다.

upgrade
Trident 의 리소스를 업그레이드하세요.

version
Trident 버전을 인쇄하세요.

글로벌 플래그

-d, --debug

디버그 출력.

-h, --help

도움말 `tridentctl` .

-k, --kubeconfig string

지정하세요 `KUBECONFIG` 로컬에서 명령을 실행하거나 한 Kubernetes 클러스터에서 다른 클러스터로 명령을 실행하는 경로입니다.

> 또는 다음을 내보낼 수 있습니다. `KUBECONFIG` 특정 Kubernetes 클러스터를 가리키고 문제를 해결하는 변수 `tridentctl` 해당 클러스터에 대한 명령입니다.

-n, --namespace string

Trident 배포의 네임스페이스.

-o, --output string

출력 형식. json|yaml|name|wide|ps 중 하나(기본값).

-s, --server string

Trident REST 인터페이스의 주소/포트.

> Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신하고 서비스하도록 구성할 수 있습니다.

명령 옵션 및 플래그

만들다

사용하다 `create` Trident 에 리소스를 추가하는 명령입니다.

```
tridentctl create [option]
```

옵션

`backend`: Trident 에 백엔드를 추가합니다.

삭제

사용하다 `delete` Trident 에서 하나 이상의 리소스를 제거하는 명령입니다.

```
tridentctl delete [option]
```

옵션

`backend`: Trident 에서 하나 이상의 스토리지 백엔드를 삭제합니다.
`snapshot` : Trident 에서 하나 이상의 볼륨 스냅샷을 삭제합니다.

`storageclass` : Trident 에서 하나 이상의 스토리지 클래스를 삭제합니다.
`volume` : Trident 에서 하나 이상의 저장 볼륨을 삭제합니다.

얻다

사용하다 `get` Trident 에서 하나 이상의 리소스를 가져오는 명령입니다.

```
tridentctl get [option]
```

옵션

`backend`: Trident 에서 하나 이상의 스토리지 백엔드를 가져옵니다.
`snapshot` : Trident 에서 하나 이상의 스냅샷을 가져옵니다.
`storageclass` : Trident 에서 하나 이상의 스토리지 클래스를 받으세요.
`volume` : Trident 에서 한 권 이상의 책을 구입하세요.

깃발

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

이미지

사용 `images` Trident 에 필요한 컨테이너 이미지 표를 인쇄하는 플래그입니다.

```
tridentctl images [flags]
```

깃발

`-h`, `--help` : 이미지에 대한 도움말.
`-v` , `--k8s-version string` : Kubernetes 클러스터의 의미적 버전.

수입량

사용하다 `import volume` 기존 볼륨을 Trident 로 가져오는 명령입니다.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

별칭

`volume`, `v`

깃발

`-f`, `--filename string` : YAML 또는 JSON PVC 파일의 경로입니다.
`-h` , `--help` : 볼륨에 대한 도움말.
`--no-manage` : PV/PVC만 생성합니다. 볼륨 수명 주기 관리를 가정하지 마세요.

설치하다

사용하다 `install` Trident 설치하기 위한 플래그입니다.

```
tridentctl install [flags]
```

깃발

`--autosupport-image string`: Autosupport Telemetry를 위한 컨테이너 이미지(기본값 "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string` : Autosupport Telemetry를 보내기 위한 프록시의 주소/포트입니다.
`--enable-node-prep` : 노드에 필요한 패키지를 설치해 보세요.
`--generate-custom-yaml` : 아무것도 설치하지 않고 YAML 파일을 생성합니다.
`-h` , `--help` : 설치에 대한 도움말.
`--http-request-timeout` : Trident 컨트롤러의 REST API에 대한 HTTP 요청 시간 초과를 재정의합니다(기본값 1분 30초).
`--image-registry string` : 내부 이미지 레지스트리의 주소/포트.
`--k8s-timeout duration` : 모든 Kubernetes 작업에 대한 시간 초과(기본값 3분 0초).
`--kubelet-dir string` : kubelet의 내부 상태의 호스트 위치(기본값 "/var/lib/kubelet").
`--log-format string` : Trident 로깅 형식(텍스트, JSON)(기본값은 "텍스트")입니다.
`--node-prep` : Trident 지정된 데이터 저장 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 합니다. 현재, **`iscsi`** 지원되는 유일한 값입니다**. OpenShift 4.19**부터 이 기능을 지원하는 최소 **Trident** 버전은 **25.06.1**입니다**.**
`--pv string` : Trident 에서 사용되는 레거시 PV의 이름으로, 이것이 존재하지 않도록 합니다(기본값은 "trident").
`--pvc string` : Trident 에서 사용하는 레거시 PVC의 이름으로, 이것이 존재하지 않도록 합니다(기본값은 "trident").
`--silence-autosupport` : NetApp 에 자동 지원 번들을 자동으로 보내지 않습니다(기본값은 true).
`--silent` : 설치 중에 대부분의 출력을 비활성화합니다.
`--trident-image string` : 설치할 Trident 이미지입니다.
`--k8s-api-qps` : Kubernetes API 요청에 대한 초당 쿼리 수(QPS) 제한(기본값 100, 선택 사항).
`--use-custom-yaml` : 설치 디렉토리에 있는 기존 YAML 파일을 사용합니다.
`--use-ipv6` : Trident 통신에 IPv6를 사용합니다.

통나무

사용 `logs` Trident 에서 로그를 인쇄하기 위한 플래그입니다.

```
tridentctl logs [flags]
```

깃발

`-a`, `--archive` : 달리 지정하지 않는 한 모든 로그를 포함하는 지원 보관소를 만듭니다.
`-h` , `--help` : 로그에 대한 도움말.
`-l` , `--log string` : Trident 로그를 표시합니다. trident|auto|trident-operator|all 중 하나(기본값은 "auto").
`--node string` : 노드 포드 로그를 수집할 Kubernetes 노드 이름입니다.
`-p` , `--previous` : 이전 컨테이너 인스턴스에 대한 로그가 있으면 가져옵니다.
`--sidecars` : 사이드카 컨테이너의 통나무를 가져오세요.

보내다

사용하다 `send` Trident 에서 리소스를 보내는 명령입니다.

```
tridentctl send [option]
```

옵션

`autosupport`: NetApp 에 Autosupport 아카이브를 보냅니다.

제거

사용 `uninstall` Trident 제거하기 위한 플래그입니다.

```
tridentctl uninstall [flags]
```

깃발

`-h, --help`: 제거에 대한 도움말.
`--silent` : 설치 제거 시 대부분의 출력을 비활성화합니다.

업데이트

사용하다 `update` Trident 에서 리소스를 수정하는 명령입니다.

```
tridentctl update [option]
```

옵션

`backend`: Trident 에서 백엔드를 업데이트합니다.

백엔드 상태 업데이트

사용하다 `update backend state` 백엔드 작업을 일시 중단하거나 재개하는 명령입니다.

```
tridentctl update backend state <backend-name> [flag]
```

고려할 사항

- TridentBackendConfig(tbc)를 사용하여 백엔드를 생성한 경우 백엔드를 사용하여 업데이트할 수 없습니다. `backend.json` 파일.
- 만약 `userState` tbc에 설정되어 있으므로 다음을 사용하여 수정할 수 없습니다. `tridentctl update backend state <backend-name> --user-state suspended/normal` 명령.
- 설정 기능을 다시 얻으려면 `userState` tbc를 통해 설정된 후 tridentctl을 통해 `userState` 해당 필드는 tbc에서 제거되어야 합니다. 이것은 다음을 사용하여 수행할 수 있습니다. `kubectl edit tbc` 명령. 후에 `userState` 필드가 제거되면 다음을 사용할 수 있습니다. `tridentctl update backend state` 변경 명령 `userState` 백엔드의.
- 사용하다 `tridentctl update backend state` 변경하려면 `userState` . 또한 업데이트할 수 있습니다 `userState` 사용 중 `TridentBackendConfig` 또는 `backend.json` 파일; 이렇게 하면 백엔드가 완전히 다시 초기화되므로 시간이 많이 걸릴 수 있습니다.

  깃발

  `-h, --help` : 백엔드 상태에 대한 도움말.
  `--user-state` : 설정 `suspended` 백엔드 작업을 일시 중지합니다. 로 설정 `normal` 백엔드 작업을 재개합니다. 설정 시 `suspended` :

- `AddVolume`그리고 `Import Volume` 일시 정지되었습니다.
- `CloneVolume`, `ResizeVolume` , `PublishVolume` , `UnPublishVolume` , `CreateSnapshot` , `GetSnapshot` , `RestoreSnapshot` , `DeleteSnapshot` , `RemoveVolume` , `GetVolumeExternal` , `ReconcileNodeAccess` 계속 이용 가능합니다.

다음을 사용하여 백엔드 상태를 업데이트할 수도 있습니다. `userState` 백엔드 구성 파일의 필드

`TridentBackendConfig` 또는 `backend.json` . 자세한 내용은 다음을 참조하세요."백엔드 관리를 위한 옵션" 그리고"kubectl을 사용하여 백엔드 관리 수행" .

예:

JSON

다음 단계에 따라 업데이트하세요. `userState` 사용하여 `backend.json` 파일:

1. 편집하다 `backend.json` 포함할 파일 `userState` 값이 '일시 중단'으로 설정된 필드입니다.
2. 다음을 사용하여 백엔드를 업데이트합니다. `tridentctl update backend` 명령 및 업데이트 경로 `backend.json` 파일.

   예: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

tbc가 적용된 후에는 다음을 사용하여 tbc를 편집할 수 있습니다. `kubectl edit <tbc-name> -n <namespace>` 명령. 다음 예제에서는 다음을 사용하여 백엔드 상태를 일시 중단으로 업데이트합니다. `userState: suspended` 옵션:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

버전

사용 `version` 버전을 인쇄하기 위한 플래그 `tridentctl` 그리고 Trident 서비스를 운영하고 있습니다.

```
tridentctl version [flags]
```

깃발

`--client`: 클라이언트 버전만 해당(서버 필요 없음).
`-h, --help` : 버전에 대한 도움말.

플러그인 지원

Tridentctl은 kubectl과 유사한 플러그인을 지원합니다. Tridentctl은 플러그인 바이너리 파일 이름이 "tridentctl-<플러그인>" 체계를 따르고 바이너리가 PATH 환경 변수에 나열된 폴더에 있는 경우 플러그인을 감지합니다. 감지된 모든 플러그인은 tridentctl 도움말의 플러그인 섹션에 나열되어 있습니다. 선택적으로 환경 변수 TRIDENTCTL_PLUGIN_PATH에 플러그인 폴더를 지정하여 검색을 제한할 수도 있습니다(예: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). 변수를 사용하면 tridenctl은 지정된 폴더에서만 검색합니다.

모니터 Trident

Trident Trident 성능을 모니터링하는 데 사용할 수 있는 일련의 Prometheus 메트릭 엔드포인트를 제공합니다.

개요

Trident 가 제공하는 측정항목을 사용하면 다음 작업을 수행할 수 있습니다.

- 트라이던트의 상태와 구성을 확인하세요. 작업이 얼마나 성공적인지, 예상대로 백엔드와 통신할 수 있는지 살펴볼 수 있습니다.
- 백엔드 사용 정보를 검토하고 백엔드에 프로비저닝된 볼륨 수와 사용된 공간의 양 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 양의 매핑을 유지합니다.
- 성과를 추적하세요. Trident 백엔드와 통신하고 작업을 수행하는 데 걸리는 시간을 살펴보세요.

기본적으로 Trident의 메트릭은 대상 포트에 노출됩니다. `8001` 에서 `/metrics` 종점. 이러한 측정항목은 Trident 가 설치되면 기본적으로 활성화됩니다.

필요한 것

- Trident 설치된 Kubernetes 클러스터.
- 프로메테우스 인스턴스. 이것은 될 수 있습니다 "컨테이너화된 Prometheus 배포" 또는 Prometheus를 실행하도록 선택할 수 있습니다. "네이티브 애플리케이션" .

1단계: Prometheus 대상 정의

Prometheus 대상을 정의하여 Trident 관리하는 백엔드, Trident가 생성하는 볼륨 등에 대한 메트릭과 정보를 수집해야 합니다. 이것 "블로그" Prometheus와 Grafana를 Trident 와 함께 사용하여 지표를 검색하는 방법을 설명합니다. 이 블로그에서는 Kubernetes 클러스터에서 Prometheus를 운영자로 실행하는 방법과 Trident 메트릭을 얻기 위해

ServiceMonitor를 만드는 방법을 설명합니다.

2단계: Prometheus ServiceMonitor 만들기

Trident 메트릭을 사용하려면 다음을 감시하는 Prometheus ServiceMonitor를 만들어야 합니다. `trident-csi` 서비스 및 청취 `metrics` 포트. ServiceMonitor 샘플은 다음과 같습니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

이 ServiceMonitor 정의는 다음에 의해 반환된 메트릭을 검색합니다. `trident-csi` 서비스를 특별히 찾습니다 `metrics` 서비스의 종료점. 결과적으로 Prometheus는 이제 Trident의 측정 항목을 이해하도록 구성되었습니다.

Trident 에서 직접 사용할 수 있는 메트릭 외에도 kubelet은 많은 것을 노출합니다. `kubelet_volume_*` 자체 메트릭 엔드포인트를 통한 메트릭. Kubelet은 연결된 볼륨, 처리하는 Pod 및 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 참조하다 "여기" .

3단계: PromQL을 사용하여 Trident 메트릭 쿼리

PromQL은 시계열이나 표 형식의 데이터를 반환하는 표현식을 만드는 데 적합합니다.

사용할 수 있는 PromQL 쿼리는 다음과 같습니다.

Trident 건강 정보 받기

- **Trident 의 HTTP 2XX 응답 비율**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Trident** 에서 상태 코드를 통해 수신한 **REST** 응답 비율

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Trident** 가 수행한 작업의 평균 지속 시간**(ms)**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Trident 사용 정보 받기

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝된 총 볼륨 공간

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

개별 볼륨 사용량 가져오기

ⓘ 이 기능은 kubelet 메트릭도 수집된 경우에만 활성화됩니다.

- 각 볼륨의 사용 공간 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Trident AutoSupport 원격 측정에 대해 알아보세요

기본적으로 Trident Prometheus 지표와 기본 백엔드 정보를 매일 NetApp 에 전송합니다.

- Trident Prometheus 메트릭 및 기본 백엔드 정보를 NetApp 으로 전송하는 것을 중지하려면 다음을 전달합니다. `--silence-autosupport` Trident 설치 중에 플래그가 표시됩니다.
- Trident 또한 다음을 통해 NetApp 지원에 컨테이너 로그를 온디맨드로 보낼 수 있습니다. `tridentctl send autosupport` . Trident 실행하여 로그를 업로드해야 합니다. 로그를 제출하기 전에 NetApp 의https://www.netapp.com/company/legal/privacy-policy/["개인정보 보호정책"^] .
- 특별히 지정하지 않으면 Trident 지난 24시간 동안의 로그를 가져옵니다.

- 로그 보존 기간을 지정할 수 있습니다. `--since` 깃발. 예를 들어: `tridentctl send autosupport --since=1h` . 이 정보는 다음을 통해 수집 및 전송됩니다. `trident-autosupport` Trident 옆에 설치된 컨테이너입니다. 컨테이너 이미지는 다음에서 얻을 수 있습니다. "Trident AutoSupport" .
- Trident AutoSupport 개인 식별 정보(PII) 또는 개인정보를 수집하거나 전송하지 않습니다. 그것은 함께 제공됩니다 "최종 사용자 사용권 계약" 이는 Trident 컨테이너 이미지 자체에는 적용되지 않습니다. NetApp의 데이터 보안 및 신뢰에 대한 노력에 대해 자세히 알아보세요. "여기" .

Trident 가 보낸 페이로드의 예는 다음과 같습니다.

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- AutoSupport 메시지는 NetApp의 AutoSupport 엔드포인트로 전송됩니다. 컨테이너 이미지를 저장하기 위해 개인 레지스트리를 사용하는 경우 다음을 사용할 수 있습니다. `--image-registry` 깃발.
- 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이것은 다음을 사용하여 수행할 수 있습니다. `tridentctl install --generate-custom-yaml` YAML 파일을 생성하고 추가하려면 `--proxy-url` 에 대한 주장 `trident-autosupport` 컨테이너에 `trident-deployment.yaml` .

Trident 메트릭 비활성화

메트릭이 보고되지 않도록 비활성화하려면 사용자 정의 YAML을 생성해야 합니다(다음을 사용). `--generate-custom-yaml` 플래그)를 편집하여 제거하세요. `--metrics` 플래그가 호출되지 않도록 `trident-main` 컨테이너.

Trident 제거

Trident 제거하는 방법은 Trident 설치하는 데 사용한 방법과 동일합니다.

이 작업에 관하여

- 업그레이드 후 발견된 버그, 종속성 문제 또는 실패하거나 완료되지 않은 업그레이드에 대한 수정이 필요한 경우 Trident 제거하고 해당 지침에 따라 이전 버전을 다시 설치해야 합니다."버전" . 이는 이전 버전으로 _다운그레이드_하는 데 권장되는 유일한 방법입니다.
- 간편한 업그레이드 및 재설치를 위해 Trident 제거해도 Trident 에서 생성된 CRD나 관련 개체는 제거되지

않습니다. Trident 와 모든 데이터를 완전히 제거해야 하는 경우 다음을 참조하세요."Trident 와 CRD를 완전히 제거하세요" .

시작하기 전에

Kubernetes 클러스터를 해제하는 경우, 제거하기 전에 Trident 에서 생성한 볼륨을 사용하는 모든 애플리케이션을 삭제해야 합니다. 이렇게 하면 PVC가 삭제되기 전에 Kubernetes 노드에서 게시 취소가 보장됩니다.

원래 설치 방법을 확인하세요

Trident 제거할 때는 설치에 사용했던 것과 동일한 방법을 사용해야 합니다. 제거하기 전에 Trident 처음 설치할 때 사용한 버전을 확인하세요.

1. 사용 `kubectl get pods -n trident` 꼬투리를 조사하기 위해서.
   - 운영자 포드가 없는 경우 Trident 설치되었습니다. `tridentctl` .
   - 운영자 포드가 있는 경우, Trident 운영자를 수동으로 사용하거나 Helm을 사용하여 Trident 설치했습니다.
2. 운영자 포드가 있는 경우 사용하세요 `kubectl describe tproc trident` Helm을 사용하여 Trident 설치되었는지 확인합니다.
   - Helm 레이블이 있으면 Helm을 사용하여 Trident 설치되었다는 의미입니다.
   - Helm 레이블이 없으면 Trident 연산자를 사용하여 Trident 수동으로 설치되었습니다.

Trident 운영자 설치 제거

Helm을 사용하거나 수동으로 Trident Operator 설치를 제거할 수 있습니다.

수동 설치 제거

운영자를 사용하여 Trident 설치한 경우 다음 중 하나를 수행하여 제거할 수 있습니다.

1. 편집하다 **`TridentOrchestrator` CR** 및 설치 제거 플래그 설정:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"uninstall":true}}'
```

   때 `uninstall` 플래그가 설정되었습니다 `true` Trident 운영자는 Trident 제거하지만 TridentOrchestrator 자체는 제거하지 않습니다. Trident 다시 설치하려면 TridentOrchestrator를 정리하고 새 TridentOrchestrator를 만들어야 합니다.

2. 삭제 **`TridentOrchestrator`** : 제거함으로써 `TridentOrchestrator` Trident 배포하는 데 사용된 CR을 사용하여 운영자에게 Trident 제거하도록 지시합니다. 운영자는 제거를 처리합니다. `TridentOrchestrator` 그리고 Trident 배포와 데몬셋을 제거하고, 설치 과정에서 생성한 Trident 포드를 삭제합니다.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Helm 설치 제거

Helm을 사용하여 Trident 설치한 경우 다음을 사용하여 제거할 수 있습니다. `helm uninstall` .

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

제거 `tridentctl` 설치

사용하다 `uninstall` 명령하다 `tridentctl` CRD 및 관련 객체를 제외한 Trident 와 연관된 모든 리소스를 제거하려면 다음을 수행합니다.

```
./tridentctl uninstall -n <namespace>
```

Docker용 Trident

배포를 위한 전제 조건

Trident 배포하려면 먼저 호스트에 필요한 프로토콜 전제 조건을 설치하고 구성해야 합니다.

요구사항을 확인하세요

- 귀하의 배포가 모든 요구 사항을 충족하는지 확인하십시오."요구 사항" .
- 지원되는 버전의 Docker가 설치되어 있는지 확인하세요. Docker 버전이 오래된 경우, "설치하거나 업데이트하세요" .

```
docker --version
```

- 호스트에 프로토콜 필수 구성 요소가 설치되고 구성되어 있는지 확인하세요.

NFS 도구

운영 체제에 맞는 명령을 사용하여 NFS 도구를 설치합니다.

RHEL 8 이상

```
sudo yum install -y nfs-utils
```

우분투

```
sudo apt-get install -y nfs-common
```

컨테이너에 볼륨을 연결할 때 오류가 발생하는 것을 방지하려면 NFS 도구를 설치한 후 작업자 노드를 재부팅하세요.

iSCSI 도구

운영 체제에 맞는 명령을 사용하여 iSCSI 도구를 설치합니다.

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. iscsi-initiator-utils 버전이 6.2.0.874-2.el7 이상인지 확인하세요.

```
rpm -q iscsi-initiator-utils
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

보장하다 `etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `iscsid` 그리고 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now iscsid multipathd
```

6. 활성화하고 시작하세요 `iscsi` :

```
sudo systemctl enable --now iscsi
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic의 경우) 또는 2.0.874-7.1ubuntu6.1 이상(focal의 경우)인지 확인하세요.

```
dpkg -l open-iscsi
```

3. 스캐닝을 수동으로 설정:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

보장하다 `etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

5. 확인하십시오 `open-iscsi` 그리고 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

NVMe 도구

운영 체제에 맞는 명령을 사용하여 NVMe 도구를 설치하세요.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 커널 버전에서 NVMe 패키지를 사용할 수 없는 경우, 노드의 커널 버전을 NVMe 패키지가 있는 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

우분투

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

FC 도구

운영 체제에 맞는 명령을 사용하여 FC 도구를 설치하세요.

- FC PV와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 다음을 지정합니다. `discard` StorageClass의 mountOption을 사용하여 인라인 공간 회수를 수행합니다. 참조하다 "Red Hat 문서" .

RHEL 8 이상

1. 다음 시스템 패키지를 설치하세요:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 다중 경로 활성화:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

보장하다 `etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipathd` 실행 중입니다:

```
sudo systemctl enable --now multipathd
```

우분투

1. 다음 시스템 패키지를 설치하세요:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. 다중 경로 활성화:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

보장하다 `etc/multipath.conf` 포함하다 `find_multipaths no` 아래에 `defaults` .

3. 확인하십시오 `multipath-tools` 활성화되어 실행 중입니다.

```
sudo systemctl status multipath-tools
```

Trident 배치

Docker용 Trident NetApp 스토리지 플랫폼을 위한 Docker 생태계와의 직접적인 통합을 제공합니다. 이는 스토리지 플랫폼에서 Docker 호스트로 스토리지 리소스를 프로비저닝하고 관리하는 기능을 지원하며, 향후 추가 플랫폼을 추가할 수 있는 프레임워크를 제공합니다.

여러 개의 Trident 인스턴스가 동일한 호스트에서 동시에 실행될 수 있습니다. 이를 통해 여러 스토리지 시스템과 스토리지 유형에 동시에 연결할 수 있으며, Docker 볼륨에 사용되는 스토리지를 사용자 정의할 수 있습니다.

필요한 것

를 참조하십시오"배포를 위한 전제 조건" . 필수 구성 요소가 충족되었는지 확인한 후 Trident 배포할 준비가 되었습니다.

Docker 관리 플러그인 방법(버전 1.13/17.03 이상)

시작하기 전에

기존 데몬 방식으로 Docker 1.13/17.03 이전의 Trident 사용한 경우, 관리형 플러그인 방식을 사용하기 전에 Trident 프로세스를 중지하고 Docker 데몬을 다시 시작해야 합니다.

1. 실행 중인 모든 인스턴스를 중지합니다.

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Docker를 다시 시작합니다.

```
systemctl restart docker
```

3. Docker Engine 17.03(새로운 1.13) 이상이 설치되어 있는지 확인하세요.

```
docker --version
```

귀하의 버전이 오래된 경우, "설치 또는 설치 업데이트" .

단계

1. 구성 파일을 만들고 다음과 같이 옵션을 지정합니다.
   - `config`: 기본 파일 이름은 다음과 같습니다. `config.json` 그러나 다음을 지정하여 원하는 이름을 사용할 수 있습니다. `config` 파일 이름이 있는 옵션입니다. 구성 파일은 다음 위치에 있어야 합니다. `/etc/netappdvp` 호스트 시스템의 디렉토리.
   - `log-level`: 로깅 수준을 지정합니다(`debug` , `info` , `warn` , `error` , `fatal`). 기본값은 `info` .
   - `debug`: 디버그 로깅을 활성화할지 여부를 지정합니다. 기본값은 false입니다. true이면 로그 수준을 재정의합니다.
     i. 구성 파일의 위치를 만듭니다.

```
sudo mkdir -p /etc/netappdvp
```

ii. 구성 파일을 만듭니다.

```
cat << EOF > /etc/netappdvp/config.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 관리되는 플러그인 시스템을 사용하여 Trident 시작합니다. 바꾸다 `<version>` 사용 중인 플러그인 버전(xxx.xx.x)을 사용하세요.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. 구성된 시스템의 저장소를 사용하기 위해 Trident 사용하기 시작합니다.

a. "firstVolume"이라는 이름의 볼륨을 생성합니다.

```
docker volume create -d netapp --name firstVolume
```

b. 컨테이너가 시작될 때 기본 볼륨을 생성합니다.

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. 볼륨 "firstVolume"을 제거합니다.

```
docker volume rm firstVolume
```

기존 방식(버전 **1.12** 이하)

시작하기 전에

1. Docker 버전이 1.10 이상인지 확인하세요.

   ```
   docker --version
   ```

   버전이 오래된 경우 설치를 업데이트하세요.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   또는, "배포에 대한 지침을 따르세요" .

2. 시스템에 NFS 및/또는 iSCSI가 구성되어 있는지 확인하세요.

단계

1. NetApp Docker Volume 플러그인을 설치하고 구성하세요.
   a. 애플리케이션을 다운로드하고 압축을 풉니다.

   ```
   wget
   https://github.com/NetApp/trident/releases/download/v25.06.0/trident-
   installer-25.06.0.tar.gz
   tar zxf trident-installer-25.06.0.tar.gz
   ```

   b. bin 경로의 위치로 이동합니다.

   ```
   sudo mv trident-installer/extras/bin/trident /usr/local/bin/
   sudo chown root:root /usr/local/bin/trident
   sudo chmod 755 /usr/local/bin/trident
   ```

   c. 구성 파일의 위치를 만듭니다.

   ```
   sudo mkdir -p /etc/netappdvp
   ```

   d. 구성 파일을 만듭니다.

   ```
   cat << EOF > /etc/netappdvp/ontap-nas.json
   ```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. 바이너리를 배치하고 구성 파일을 만든 후, 원하는 구성 파일을 사용하여 Trident 데몬을 시작합니다.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

> 특별히 지정하지 않으면 볼륨 드라이버의 기본 이름은 "netapp"입니다.

데몬이 시작된 후 Docker CLI 인터페이스를 사용하여 볼륨을 만들고 관리할 수 있습니다.

3. 볼륨을 생성합니다.

```
docker volume create -d netapp --name trident_1
```

4. 컨테이너를 시작할 때 Docker 볼륨을 프로비저닝합니다.

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Docker 볼륨 제거:

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

시스템 시작 시 **Trident** 시작

systemd 기반 시스템에 대한 샘플 단위 파일은 다음에서 찾을 수 있습니다.
`contrib/trident.service.example` Git 저장소에 있습니다. RHEL에서 파일을 사용하려면 다음을 수행하세요.

1. 파일을 올바른 위치에 복사하세요.

   두 개 이상의 인스턴스를 실행하는 경우 단위 파일에 고유한 이름을 사용해야 합니다.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. 파일을 편집하여 드라이버 이름과 구성 파일 경로(9번째 줄)에 맞게 설명(2번째 줄)을 변경하고 사용자 환경을 반영합니다.
3. 변경 사항을 적용하려면 systemd를 다시 로드하세요.

   ```
   systemctl daemon-reload
   ```

4. 서비스를 활성화합니다.

   이 이름은 파일 이름을 무엇으로 지정했는지에 따라 달라집니다. `/usr/lib/systemd/system` 예배 규칙서.

   ```
   systemctl enable trident
   ```

5. 서비스를 시작합니다.

   ```
   systemctl start trident
   ```

6. 상태를 확인하세요.

   ```
   systemctl status trident
   ```

> 단위 파일을 수정할 때마다 다음을 실행하세요. `systemctl daemon-reload` 변경 사항을 인식하도록 명령합니다.

Trident 업그레이드 또는 제거

사용 중인 볼륨에 영향을 주지 않고 Docker용 Trident 안전하게 업그레이드할 수 있습니다. 업그레이드 프로세스 동안 잠시 시간이 걸립니다. `docker volume` 플러그인에 대한 명령은 성공하지 못하고, 플러그인이 다시 실행될 때까지 애플리케이션은 볼륨을 마운트할 수 없습니다. 대부분의 경우 이는 몇 초 만에 끝납니다.

치받이

Docker용 Trident 업그레이드하려면 아래 단계를 수행하세요.

단계

1. 기존 볼륨을 나열하세요:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. 플러그인을 비활성화하세요:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 플러그인을 업그레이드하세요:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> Trident 의 18.01 릴리스는 nDVP를 대체합니다. 직접 업그레이드해야 합니다. `netapp/ndvp-plugin` 이미지를 `netapp/trident-plugin` 영상.

4. 플러그인을 활성화하세요:

```
docker plugin enable netapp:latest
```

5. 플러그인이 활성화되어 있는지 확인하세요.

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. 볼륨이 표시되는지 확인하세요.

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

Trident 의 이전 버전(20.10 이전)에서 Trident 20.10 이상으로 업그레이드하는 경우 오류가 발생할 수 있습니다. 자세한 내용은 다음을 참조하세요."알려진 문제" . 오류가 발생하면 먼저 플러그인을 비활성화한 다음 플러그인을 제거한 다음 추가 구성 매개변수를 전달하여 필요한 Trident 버전을 설치해야 합니다. `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

제거

Docker용 Trident 제거하려면 아래 단계를 수행하세요.

단계

1. 플러그인이 생성한 모든 볼륨을 제거합니다.
2. 플러그인을 비활성화하세요:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 플러그인을 제거하세요:

```
docker plugin rm netapp:latest
```

볼륨 작업

표준을 사용하여 볼륨을 쉽게 생성, 복제 및 제거할 수 있습니다. `docker volume` 필요한 경우 Trident 드라이버 이름을 지정하여 명령을 실행합니다.

볼륨을 생성합니다

- 기본 이름을 사용하여 드라이버로 볼륨을 생성합니다.

```
docker volume create -d netapp --name firstVolume
```

- 특정 Trident 인스턴스로 볼륨을 생성합니다.

```
docker volume create -d ntap_bronze --name bronzeVolume
```

아무것도 지정하지 않으면"옵션" 드라이버의 기본값이 사용됩니다.

- 기본 볼륨 크기를 재정의합니다. 드라이버를 사용하여 20GiB 볼륨을 생성하는 다음 예를 참조하세요.

```
docker volume create -d netapp --name my_vol --opt size=20G
```

볼륨 크기는 선택적 단위(예: 10G, 20GB, 3TiB)가 포함된 정수 값을 포함하는 문자열로 표현됩니다. 단위를 지정하지 않으면 기본값은 G입니다. 크기 단위는 2의 거듭제곱(B, KiB, MiB, GiB, TiB) 또는 10의 거듭제곱(B, KB, MB, GB, TB)으로 표현할 수 있습니다. 약어 단위는 2의 거듭제곱을 사용합니다(G = GiB, T = TiB, …).

볼륨 제거

- 다른 Docker 볼륨과 마찬가지로 볼륨을 제거합니다.

```
docker volume rm firstVolume
```

사용 시 `solidfire-san` 드라이버의 경우 위의 예에서는 볼륨을 삭제하고 제거합니다.

Docker용 Trident 업그레이드하려면 아래 단계를 수행하세요.

볼륨 복제

사용 시 `ontap-nas` , `ontap-san` , `solidfire-san` , 그리고 `gcp-cvs storage drivers` Trident 볼륨을 복제할 수 있습니다. 사용 시 `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버, 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새로운 스냅샷이 생성됩니다.

- 볼륨을 검사하여 스냅샷을 열거합니다.

```
docker volume inspect <volume_name>
```

- 기존 볼륨에서 새 볼륨을 만듭니다. 그러면 새로운 스냅샷이 생성됩니다.

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- 볼륨의 기존 스냅샷에서 새 볼륨을 만듭니다. 이렇게 하면 새로운 스냅샷이 생성되지 않습니다.

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

예

```
docker volume inspect firstVolume

[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

외부에서 생성된 볼륨에 액세스

Trident 사용하는 컨테이너는 파티션이 없고 파일 시스템이 Trident 에서 지원되는 경우에만 외부에서 생성된 블록 장치(또는 해당 복제본)에 액세스할 수 있습니다(예: `ext4` -포맷됨 `/dev/sdc1` Trident 통해서는 접근할 수 없습니다.)

드라이버별 볼륨 옵션

각 스토리지 드라이버에는 서로 다른 옵션 세트가 있으며, 볼륨을 생성할 때 이를 지정하여 결과를 사용자 정의할 수 있습니다. 구성된 스토리지 시스템에 적용되는 옵션은 아래를 참조하세요.

볼륨 생성 작업 중에 이러한 옵션을 사용하는 것은 간단합니다. 옵션과 값을 제공합니다. `-o` CLI 작업 중 연산자. 이는 JSON 구성 파일의 동등한 값을 재정의합니다.

ONTAP 볼륨 옵션

NFS, iSCSI 및 FC에 대한 볼륨 생성 옵션은 다음과 같습니다.

옵션	설명
`size`	볼륨 크기는 기본적으로 1GiB입니다.
`spaceReserve`	볼륨을 얇거나 두껍게 프로비저닝하고 기본적으로 얇음이 지정됩니다. 유효한 값은 다음과 같습니다. `none` (씬 프로비저닝) 및 `volume` (두꺼운 식량).
`snapshotPolicy`	이렇게 하면 스냅샷 정책이 원하는 값으로 설정됩니다. 기본값은 `none` 즉, 볼륨에 대한 스냅샷이 자동으로 생성되지 않습니다. 스토리지 관리자가 수정하지 않는 한, 모든 ONTAP 시스템에는 6시간마다, 2일마다, 2주마다 스냅샷을 생성하고 보관하는 "기본값"이라는 정책이 존재합니다. 스냅샷에 보존된 데이터는 다음을 탐색하여 복구할 수 있습니다. `.snapshot` 볼륨 내의 모든 디렉토리에 있는 디렉토리.
`snapshotReserve`	이렇게 하면 스냅샷 예약이 원하는 비율로 설정됩니다. 기본값은 값이 없습니다. 즉, 스냅샷 정책을 선택한 경우 ONTAP 스냅샷 예약(일반적으로 5%)을 선택하고, 스냅샷 정책이 없는 경우 0%를 선택합니다. 모든 ONTAP 백엔드의 구성 파일에서 기본 snapshotReserve 값을 설정할 수 있으며, ontap-nas-economy를 제외한 모든 ONTAP 백엔드에 대해 볼륨 생성 옵션으로 사용할 수 있습니다.
`splitOnClone`	볼륨을 복제할 때 ONTAP 은 복제본을 부모 볼륨에서 즉시 분리합니다. 기본값은 `false` . 볼륨 복제의 일부 사용 사례에서는 복제본을 생성하자마자 부모 볼륨에서 즉시 분리하는 것이 가장 좋습니다. 저장 효율성을 높일 기회가 거의 없기 때문입니다. 예를 들어, 빈 데이터베이스를 복제하면 시간은 크게 절약할 수 있지만 저장 공간은 거의 절약할 수 없으므로 복제본을 즉시 분할하는 것이 가장 좋습니다.

옵션	설명
encryption	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. false . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .
tieringPolicy	볼륨에 사용할 계층화 정책을 설정합니다. 이는 데이터가 비활성화(콜드)될 때 클라우드 계층으로 이동할지 여부를 결정합니다.

다음 추가 옵션은 NFS에만 적용됩니다.

옵션	설명
unixPermissions	이는 볼륨 자체에 대한 권한 집합을 제어합니다. 기본적으로 권한은 다음과 같이 설정됩니다. `---rwxr-xr-x 또는 숫자 표기법으로 0755, root 소유자가 될 것입니다. 텍스트나 숫자 형식 모두 작동합니다.
snapshotDir	이것을 설정하려면 true 만들 것이다 .snapshot 볼륨에 액세스하는 클라이언트가 볼 수 있는 디렉토리입니다. 기본값은 false 즉, 가시성이 있다는 의미입니다. .snapshot 디렉토리는 기본적으로 비활성화되어 있습니다. 예를 들어 공식 MySQL 이미지와 같은 일부 이미지는 예상대로 작동하지 않습니다. .snapshot 디렉토리가 표시됩니다.
exportPolicy	볼륨에 사용할 내보내기 정책을 설정합니다. 기본값은 default .
securityStyle	볼륨에 액세스하는 데 사용할 보안 스타일을 설정합니다. 기본값은 unix . 유효한 값은 다음과 같습니다. unix 그리고 mixed .

다음 추가 옵션은 iSCSI에만 적용됩니다.

옵션	설명
fileSystemType	iSCSI 볼륨을 포맷하는 데 사용되는 파일 시스템을 설정합니다. 기본값은 ext4 . 유효한 값은 다음과 같습니다. ext3 , ext4 , 그리고 xfs .

옵션	설명
spaceAllocation	이것을 설정하려면 false LUN의 공간 할당 기능을 끕니다. 기본값은 true 즉, ONTAP 볼륨 공간이 부족하고 볼륨의 LUN이 쓰기를 허용할 수 없을 때 호스트에 알립니다. 이 옵션을 사용하면 호스트가 데이터를 삭제할 때 ONTAP 자동으로 공간을 회수할 수도 있습니다.

예시

아래의 예를 참조하세요.

- 10GiB 볼륨을 생성합니다.

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 스냅샷으로 100GiB 볼륨을 만듭니다.

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- setUID 비트가 활성화된 볼륨을 생성합니다.

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

최소 볼륨 크기는 20MiB입니다.

스냅샷 예약이 지정되지 않고 스냅샷 정책이 none Trident 0%의 스냅샷 예비금을 사용합니다.

- 스냅샷 정책 및 스냅샷 예약이 없는 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- 스냅샷 정책이 없고 사용자 지정 스냅샷 예약이 10%인 볼륨을 만듭니다.

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- 스냅샷 정책과 10%의 사용자 지정 스냅샷 예약으로 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- 스냅샷 정책으로 볼륨을 생성하고 ONTAP의 기본 스냅샷 예약(일반적으로 5%)을 수락합니다.

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Element 소프트웨어 볼륨 옵션

Element 소프트웨어 옵션은 볼륨과 관련된 크기 및 서비스 품질(QoS) 정책을 보여줍니다. 볼륨이 생성되면 해당 볼륨과 연관된 QoS 정책이 다음을 사용하여 지정됩니다. `-o type=service_level` 명명법.

Element 드라이버를 사용하여 QoS 서비스 수준을 정의하는 첫 번째 단계는 최소한 하나의 유형을 만들고 구성 파일에서 이름과 연관된 최소, 최대 및 버스트 IOPS를 지정하는 것입니다.

Element 소프트웨어의 기타 볼륨 생성 옵션은 다음과 같습니다.

옵션	설명
`size`	볼륨 크기는 기본적으로 1GiB 또는 구성 항목 … "defaults": {"size": "5G"}입니다.
`blocksize`	512 또는 4096을 사용하고, 기본값은 512 또는 DefaultBlockSize 구성 항목입니다.

예

QoS 정의가 포함된 다음 샘플 구성 파일을 참조하세요.

```
{
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

위의 구성에서는 Bronze, Silver, Gold의 세 가지 정책 정의가 있습니다. 이러한 이름은 임의적입니다.

- 10GiB Gold 볼륨을 생성합니다.

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 100GiB Bronze 볼륨을 생성합니다.

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

로그 수집

문제 해결에 도움이 되는 로그를 수집할 수 있습니다. 로그를 수집하는 데 사용하는 방법은 Docker 플러그인을 실행하는 방법에 따라 달라집니다.

문제 해결을 위한 로그 수집

단계

1. 권장되는 관리 플러그인 방식(예: 사용)을 사용하여 Trident 실행하는 경우 `docker plugin` 명령)을 다음과 같이 확인하세요.

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

   표준 로깅 수준을 사용하면 대부분의 문제를 진단할 수 있습니다. 그래도 충분하지 않다면 디버그 로깅을 활성화할 수 있습니다.

2. 디버그 로깅을 활성화하려면 디버그 로깅이 활성화된 플러그인을 설치하세요.

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   또는 플러그인이 이미 설치된 경우 디버그 로깅을 활성화합니다.

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. 호스트에서 바이너리 자체를 실행하는 경우 로그는 호스트에서 사용할 수 있습니다. `/var/log/netappdvp` 예배 규칙서. 디버그 로깅을 활성화하려면 다음을 지정하세요. `-debug` 플러그인을 실행할 때.

일반적인 문제 해결 팁

- 새로운 사용자가 겪는 가장 흔한 문제는 플러그인 초기화를 방해하는 잘못된 구성입니다. 이런 일이 발생하면 플러그인을 설치하거나 활성화하려고 할 때 다음과 같은 메시지가 표시될 가능성이 높습니다.

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  이는 플러그인이 시작되지 않았음을 의미합니다. 다행히도 이 플러그인은 포괄적인 로깅 기능을 내장하고 있어, 발생할 수 있는 대부분의 문제를 진단하는 데 도움이 됩니다.

- 컨테이너에 PV를 장착하는 데 문제가 있는 경우 다음을 확인하십시오. `rpcbind` 설치되어 실행 중입니다. 호스트 OS에 필요한 패키지 관리자를 사용하여 확인하십시오. `rpcbind` 실행 중입니다. rpcbind 서비스의 상태를 확인하려면 다음을 실행하세요. `systemctl status rpcbind` 또는 이에 상응하는 것.

여러 Trident 인스턴스 관리

여러 스토리지 구성을 동시에 사용하려면 Trident 의 여러 인스턴스가 필요합니다. 여러 인스턴스의 핵심은 다음을 사용하여 서로 다른 이름을 지정하는 것입니다. `--alias` 컨테이너화된 플러그인을 사용한 옵션 또는 `--volume-driver` 호스트에서 Trident 인스턴스화할 때의 옵션입니다.

Docker 관리 플러그인(버전 1.13/17.03 이상)에 대한 단계

1. 별칭과 구성 파일을 지정하여 첫 번째 인스턴스를 시작합니다.

   ```
   docker plugin install --grant-all-permissions --alias silver
   netapp/trident-plugin:21.07 config=silver.json
   ```

2. 다른 별칭과 구성 파일을 지정하여 두 번째 인스턴스를 시작합니다.

   ```
   docker plugin install --grant-all-permissions --alias gold
   netapp/trident-plugin:21.07 config=gold.json
   ```

3. 드라이버 이름으로 별칭을 지정하여 볼륨을 생성합니다.

   예를 들어, 금 거래량의 경우:

   ```
   docker volume create -d gold --name ntapGold
   ```

   예를 들어, 은의 거래량은 다음과 같습니다.

   ```
   docker volume create -d silver --name ntapSilver
   ```

기존 버전(1.12 이하)에 대한 단계

1. 사용자 정의 드라이버 ID를 사용하여 NFS 구성으로 플러그인을 실행합니다.

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 사용자 정의 드라이버 ID를 사용하여 iSCSI 구성으로 플러그인을 실행합니다.

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. 각 드라이버 인스턴스에 대한 Docker 볼륨 프로비저닝:

   예를 들어, NFS의 경우:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   예를 들어 iSCSI의 경우:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

스토리지 구성 옵션

Trident 구성에 사용할 수 있는 구성 옵션을 확인하세요.

글로벌 구성 옵션

이러한 구성 옵션은 사용되는 스토리지 플랫폼에 관계없이 모든 Trident 구성에 적용됩니다.

옵션	설명	예
`version`	구성 파일 버전 번호	`1`
`storageDriverName`	저장 드라이버 이름	`ontap-nas`, `ontap-san` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `solidfire-san`
`storagePrefix`	볼륨 이름에 대한 선택적 접두사입니다. 기본: `netappdvp_` .	`staging_`

옵션	설명	예
`limitVolumeSize`	볼륨 크기에 대한 선택적 제한. 기본값: ""(강제되지 않음)	`10g`

사용하지 마십시오 `storagePrefix` (기본값 포함) Element 백엔드에 대한 것입니다. 기본적으로, `solidfire-san` 드라이버는 이 설정을 무시하고 접두사를 사용하지 않습니다. NetApp Docker 볼륨 매핑에 특정 테넌트 ID를 사용하거나 이름 변경이 사용된 경우 Docker 버전, 드라이버 정보 및 Docker의 원시 이름으로 채워진 속성 데이터를 사용할 것을 권장합니다.

볼륨을 생성할 때마다 옵션을 지정하지 않아도 되도록 기본 옵션을 사용할 수 있습니다. 그만큼 `size` 모든 컨트롤러 유형에 옵션이 제공됩니다. 기본 볼륨 크기를 설정하는 방법에 대한 예는 ONTAP 구성 섹션을 참조하세요.

옵션	설명	예
`size`	새 볼륨에 대한 선택적인 기본 크기입니다. 기본: `1G`	`10G`

ONTAP 구성

위의 글로벌 구성 값 외에도 ONTAP 사용할 때 다음과 같은 최상위 옵션을 사용할 수 있습니다.

옵션	설명	예
`managementLIF`	ONTAP 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다.	`10.0.0.1`
`dataLIF`	프로토콜 LIF의 IP 주소. • ONTAP NAS 드라이버*: NetApp 다음을 지정하는 것을 권장합니다. `dataLIF` . 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. • ONTAP SAN 드라이버*: iSCSI 또는 FC를 지정하지 마세요. Trident 사용"ONTAP 선택적 LUN 맵" 다중 경로 세션을 설정하는 데 필요한 iSCSI 또는 FC LIF를 검색합니다. 경고가 생성됩니다. `dataLIF` 명확하게 정의되어 있습니다.	`10.0.0.2`

옵션	설명	예
`svm`	사용할 스토리지 가상 머신(관리 LIF가 클러스터 LIF인 경우 필수)	`svm_nfs`
`username`	저장 장치에 연결할 사용자 이름	`vsadmin`
`password`	저장장치에 연결하기 위한 비밀번호	`secret`
`aggregate`	프로비저닝을 위한 집계(선택 사항, 설정된 경우 SVM에 할당해야 함). 를 위해 `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. SVM에 할당된 모든 집계는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다.	`aggr1`
`limitAggregateUsage`	선택 사항, 사용량이 이 백분율을 초과하면 프로비저닝이 실패합니다.	`75%`
`nfsMountOptions`	NFS 마운트 옵션에 대한 세부적인 제어; 기본값은 "-o nfsvers=3"입니다. 다음에 한해서만 사용 가능합니다. **`ontap-nas`** 그리고 **`ontap-nas-economy`** 운전자. "NFS 호스트 구성 정보는 여기에서 확인하세요." .	`-o nfsvers=4`
`igroupName`	Trident 노드별로 생성하고 관리합니다. `igroups` ~처럼 `netappdvp` . 이 값은 변경하거나 생략할 수 없습니다. 다음에 한해서만 사용 가능합니다. **`ontap-san`** 운전사.	`netappdvp`
`limitVolumeSize`	요청 가능한 최대 볼륨 크기.	`300g`
`qtreesPerFlexvol`	FlexVol 당 최대 qtree는 [50, 300] 범위 내에 있어야 하며 기본값은 200입니다. *에 대한 `ontap-nas-economy` 드라이버, 이 옵션을 사용하면 FlexVol*당 최대 qtree 수를 사용자 지정할 수 있습니다.	`300`

옵션	설명	예
sanType	지원됨 **ontap-san** 운전자만. 선택에 사용 iscsi iSCSI의 경우, nvme NVMe/TCP 또는 fcp FC(Fibre Channel)를 통한 SCSI의 경우.	`iscsi`비어있는 경우
limitVolumePoolSize	지원됨 **ontap-san-economy** 그리고 **ontap-san-economy** 운전자만 해당. ONTAP ontap-nas-economy 및 ontap-SAN-economy 드라이버에서 FlexVol 크기를 제한합니다.	300g

볼륨을 생성할 때마다 지정하지 않아도 되도록 기본 옵션을 사용할 수 있습니다.

옵션	설명	예
spaceReserve	공간 예약 모드; none (씬 프로비저닝) 또는 volume (두꺼운)	none
snapshotPolicy	사용할 스냅샷 정책, 기본값은 다음과 같습니다. none	none
snapshotReserve	스냅샷 예약 비율, ONTAP 기본값을 적용하려면 기본값은 ""입니다.	10
splitOnClone	생성 시 부모로부터 복제본을 분할합니다. 기본값은 다음과 같습니다. false	false
encryption	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화합니다. 기본값은 다음과 같습니다. false . 이 옵션을 사용하려면 클러스터에서 NVE에 대한 라이선스를 받고 활성화해야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식" .	true
unixPermissions	프로비저닝된 NFS 볼륨에 대한 NAS 옵션, 기본값은 다음과 같습니다. 777	777
snapshotDir	NAS 옵션으로 액세스 가능 .snapshot 예배 규칙서.	NFSv4의 경우 "true", NFSv3의 경우 "false"
exportPolicy	NFS 내보내기 정책에 사용할 NAS 옵션은 기본적으로 다음과 같습니다. default	default

옵션	설명	예
securityStyle	프로비저닝된 NFS 볼륨에 액세스하기 위한 NAS 옵션입니다. NFS 지원 mixed 그리고 unix 보안 스타일. 기본값은 unix .	unix
fileSystemType	파일 시스템 유형을 선택하는 SAN 옵션, 기본값은 다음과 같습니다. ext4	xfs
tieringPolicy	사용할 계층화 정책, 기본값은 다음과 같습니다. none .	none

스케일링 옵션

그만큼 ontap-nas 그리고 ontap-san 드라이버는 각 Docker 볼륨에 대해 ONTAP FlexVol 생성합니다. ONTAP 클러스터 노드당 최대 1000개의 FlexVol을 지원하며 클러스터 최대 FlexVol 볼륨은 12,000개입니다. Docker 볼륨 요구 사항이 해당 제한 사항에 맞는 경우 ontap-nas FlexVols가 제공하는 Docker 볼륨 단위 스냅샷 및 복제와 같은 추가 기능으로 인해 드라이버가 선호되는 NAS 솔루션입니다.

FlexVol 제한으로 수용할 수 있는 것보다 더 많은 Docker 볼륨이 필요한 경우 다음을 선택하십시오. ontap-nas-economy 또는 ontap-san-economy 운전사.

그만큼 ontap-nas-economy 드라이버는 자동으로 관리되는 FlexVol 볼륨 풀 내에서 ONTAP Qtree로 Docker 볼륨을 생성합니다. Qtree는 일부 기능을 희생하더라도 클러스터 노드당 최대 100,000개, 클러스터당 최대 2,400,000개까지 확장할 수 있는 훨씬 더 큰 확장성을 제공합니다. 그만큼 ontap-nas-economy 드라이버는 Docker 볼륨 단위의 스냅샷이나 복제를 지원하지 않습니다.

> 그만큼 ontap-nas-economy Docker Swarm은 여러 노드에 걸쳐 볼륨 생성을 조정하지 않으므로 해당 드라이버는 현재 Docker Swarm에서 지원되지 않습니다.

그만큼 ontap-san-economy 드라이버는 자동으로 관리되는 FlexVol 볼륨의 공유 풀 내에서 ONTAP LUN으로 Docker 볼륨을 생성합니다. 이렇게 하면 각 FlexVol 단 하나의 LUN에만 제한되지 않고 SAN 워크로드에 대한 확장성이 향상됩니다. 스토리지 어레이에 따라 ONTAP 클러스터당 최대 16384개의 LUN을 지원합니다. 볼륨이 그 아래의 LUN이므로 이 드라이버는 Docker 볼륨 단위 스냅샷과 복제를 지원합니다.

선택하세요 ontap-nas-flexgroup 수십억 개의 파일을 저장할 수 있는 페타바이트 범위까지 확장 가능한 단일 볼륨에 대한 병렬 처리를 높이는 드라이버입니다. FlexGroups의 이상적인 사용 사례로는 AI/ML/DL, 빅데이터 및 분석, 소프트웨어 빌드, 스트리밍, 파일 저장소 등이 있습니다. Trident FlexGroup 볼륨을 프로비저닝할 때 SVM에 할당된 모든 집계를 사용합니다. Trident 의 FlexGroup 지원에는 다음과 같은 고려 사항이 있습니다.

- ONTAP 버전 9.2 이상이 필요합니다.
- 이 글을 쓰는 시점에서 FlexGroups는 NFS v3만 지원합니다.
- SVM에 대해 64비트 NFSv3 식별자를 활성화하는 것이 좋습니다.
- 권장되는 최소 FlexGroup 멤버/볼륨 크기는 100GiB입니다.
- FlexGroup 볼륨에서는 복제가 지원되지 않습니다.

FlexGroups 및 FlexGroups에 적합한 워크로드에 대한 정보는 다음을 참조하세요. "NetApp FlexGroup 볼륨 모범

사례 및 구현 가이드" .

동일한 환경에서 고급 기능과 대규모 기능을 얻으려면 Docker Volume Plugin의 여러 인스턴스를 실행할 수 있습니다. `ontap-nas` 그리고 또 다른 사용 `ontap-nas-economy` .

Trident 용 사용자 정의 ONTAP 역할

Trident 에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용하지 않아도 되도록 최소한의 권한으로 ONTAP 클러스터 역할을 만들 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident 사용자가 만든 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

참조하다"Trident 사용자 정의 역할 생성기" Trident 사용자 정의 역할 생성에 대한 자세한 내용은 다음을 참조하세요.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 만듭니다.

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Trident 사용자에 대한 사용자 이름을 만듭니다.

```
security login create -username <user_name\> -application ontapi
-authmethod password -role <name_of_role_in_step_1\> -vserver <svm_name\>
-comment "user_description"
security login create -username <user_name\> -application http -authmethod
password -role <name_of_role_in_step_1\> -vserver <svm_name\> -comment
"user_description"
```

3. 사용자에게 역할을 매핑합니다.

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

시스템 관리자 사용

ONTAP 시스템 관리자에서 다음 단계를 수행합니다.

1. 사용자 정의 역할 만들기:
   a. 클러스터 수준에서 사용자 지정 역할을 만들려면 *클러스터 > 설정*을 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 만들려면 저장소 **>** 저장소 **VM > `required SVM` >** 설정 **>** 사용자 및 역할.

   b. 사용자 및 역할 옆에 있는 화살표 아이콘(→)을 선택합니다.
   c. *역할*에서 *+추가*를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 *저장*을 클릭합니다.
2. * Trident 사용자에게 역할 매핑*: + 사용자 및 역할 페이지에서 다음 단계를 수행합니다.
   a. 사용자 아래에 있는 추가 아이콘 *+*을 선택합니다.
   b. 필요한 사용자 이름을 선택하고, 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭하세요.

자세한 내용은 다음 페이지를 참조하세요.

- "ONTAP 관리를 위한 사용자 정의 역할"또는"사용자 정의 역할 정의"
- "역할 및 사용자 작업"

ONTAP 구성 파일 예

<code>ontap-nas</code> 드라이버에 대한 NFS 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "defaults": {
    "size": "10G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

<code>ontap-nas-flexgroup</code> 드라이버에 대한 NFS 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "defaults": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

<code>ontap-nas-economy</code> 드라이버에 대한 **NFS** 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
```

<code>ontap-san</code> 드라이버에 대한 **iSCSI** 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

<code>ontap-san-economy</code> 드라이버에 대한 **NFS** 예제

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

<code>ontap-san</code> 드라이버에 대한 NVMe/TCP 예제

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

<code>ontap-san</code> 드라이버에 대한 FC를 통한 SCSI 예제

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "useREST": true
}
```

요소 소프트웨어 구성

Element 소프트웨어(NetApp HCI/ SolidFire)를 사용하는 경우 글로벌 구성 값 외에도 다음 옵션을 사용할 수 있습니다.

옵션	설명	예
`Endpoint`	https://<로그인>:<비밀번호>@<mvip>/json-rpc/<요소 버전>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	iSCSI IP 주소 및 포트	10.0.0.7:3260
`TenantName`	SolidFireF 사용할 테넌트(찾을 수 없는 경우 생성됨)	`docker`

옵션	설명	예
`InitiatorIFace`	iSCSI 트래픽을 기본 인터페이스가 아닌 인터페이스로 제한할 때 인터페이스를 지정합니다.	`default`
`Types`	QoS 사양	아래 예를 참조하세요
`LegacyNamePrefix`	업그레이드된 Trident 설치를 위한 접두사입니다. 1.3.2 이전 버전의 Trident 사용하고 기존 볼륨으로 업그레이드를 수행한 경우 볼륨 이름 방식을 통해 매핑된 이전 볼륨에 액세스하려면 이 값을 설정해야 합니다.	`netappdvp-`

그만큼 `solidfire-san` 드라이버가 Docker Swarm을 지원하지 않습니다.

Element 소프트웨어 구성 파일 예시

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
  "SVIP": "10.0.0.7:3260",
  "TenantName": "docker",
  "InitiatorIFace": "default",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

알려진 문제 및 제한 사항

Docker와 함께 Trident 사용할 때 알려진 문제와 제한 사항에 대한 정보를 찾아보세요.

Trident Docker Volume Plugin을 이전 버전에서 **20.10** 이상으로 업그레이드하면 해당 파일이나 디렉토리가 없다는 오류와 함께 업그레이드가 실패합니다.

해결 방법

1. 플러그인을 비활성화합니다.

```
docker plugin disable -f netapp:latest
```

2. 플러그인을 제거하세요.

```
docker plugin rm -f netapp:latest
```

3. 추가 기능을 제공하여 플러그인을 다시 설치합니다. `config` 매개변수.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

볼륨 이름은 최소 2자 이상이어야 합니다.

이는 Docker 클라이언트의 제한 사항입니다. 클라이언트는 단일 문자 이름을 Windows 경로로 해석합니다. "버그 25773을 참조하세요" .

Docker Swarm에는 Trident 모든 스토리지 및 드라이버 조합을 지원하지 못하게 하는 특정 동작이 있습니다.

- Docker Swarm은 현재 볼륨 ID 대신 볼륨 이름을 고유한 볼륨 식별자로 사용합니다.
- 볼륨 요청은 Swarm 클러스터의 각 노드에 동시에 전송됩니다.
- 볼륨 플러그인(Trident 포함)은 Swarm 클러스터의 각 노드에서 독립적으로 실행되어야 합니다. ONTAP 의 작동 방식과 `ontap-nas` 그리고 `ontap-san` 운전자가 기능하는 한, 이러한 제한 내에서 작동할 수 있는 유일한 운전자가 운전자입니다.

나머지 드라이버는 명확한 "승자" 없이 단일 요청에 대해 많은 수의 볼륨이 생성되는 결과를 초래할 수 있는 경쟁 조건과 같은 문제의 영향을 받습니다. 예를 들어 Element에는 볼륨의 이름은 같지만 ID는 다를 수 있는 기능이 있습니다.

NetApp Docker 팀에 피드백을 제공했지만 향후 대책에 대한 내용은 없습니다.

FlexGroup 이 프로비저닝되는 경우, 두 번째 FlexGroup 에 프로비저닝되는 FlexGroup 과 공통된 집계가 하나 이상 있는 경우 ONTAP 두 번째 FlexGroup 프로비저닝하지 않습니다.

모범 사례 및 권장 사항

전개

Trident 배포할 때 여기에 나열된 권장 사항을 사용하세요.

전용 네임스페이스에 배포

"네임스페이스"서로 다른 애플리케이션 간에 관리적 분리를 제공하고 리소스 공유에 대한 장벽이 됩니다. 예를 들어, 한 네임스페이스의 PVC는 다른 네임스페이스에서 사용될 수 없습니다. Trident Kubernetes 클러스터의 모든 네임스페이스에 PV 리소스를 제공하고 결과적으로 권한이 상승된 서비스 계정을 활용합니다.

또한, Trident 포드에 접근하면 사용자가 스토리지 시스템 자격 증명 및 기타 민감한 정보에 접근할 수 있습니다. 애플리케이션 사용자와 관리 애플리케이션이 Trident 객체 정의나 포드 자체에 액세스할 수 없도록 하는 것이 중요합니다.

할당량과 범위 제한을 사용하여 저장소 소비를 제어합니다.

쿠버네티스에는 두 가지 기능이 있는데, 이 두 가지를 결합하면 애플리케이션의 리소스 소비를 제한하는 강력한 메커니즘을 제공합니다. 그만큼 "저장 할당량 메커니즘" 관리자가 네임스페이스별로 글로벌 및 스토리지 클래스별 용량 및 개체 수 소비 제한을 구현할 수 있도록 합니다. 또한, 다음을 사용하여 "범위 제한" 요청이 프로비저너로 전달되기 전에 PVC 요청이 최소값과 최대값 내에 있는지 확인합니다.

이러한 값은 네임스페이스별로 정의됩니다. 즉, 각 네임스페이스에는 리소스 요구 사항에 맞는 값이 정의되어 있어야 합니다. 여기에서 정보를 확인하세요 "할당량을 활용하는 방법" .

스토리지 구성

NetApp 포트폴리오의 각 스토리지 플랫폼은 컨테이너화 여부와 관계없이 애플리케이션에 도움이 되는 고유한 기능을 갖추고 있습니다.

플랫폼 개요

Trident ONTAP 및 Element와 함께 작동합니다. 모든 애플리케이션과 시나리오에 더 적합한 플랫폼은 없습니다. 그러나 플랫폼을 선택할 때는 애플리케이션의 요구 사항과 장치를 관리하는 팀을 고려해야 합니다.

사용하는 프로토콜에 맞는 호스트 운영 체제의 기본 모범 사례를 따라야 합니다. 선택적으로, 특정 애플리케이션에 맞게 스토리지를 최적화하기 위해 백엔드, 스토리지 클래스 및 PVC 설정과 함께 애플리케이션 모범 사례를 통합하는 것을 고려할 수도 있습니다.

ONTAP 및 Cloud Volumes ONTAP 모범 사례

Trident 위한 ONTAP 및 Cloud Volumes ONTAP 구성에 대한 모범 사례를 알아보세요.

다음 권장 사항은 Trident 에서 동적으로 프로비저닝되는 볼륨을 사용하는 컨테이너화된 워크로드에 대해 ONTAP 구성하기 위한 지침입니다. 각 항목은 귀하의 환경에 적합한지 여부를 고려하여 평가해야 합니다.

Trident 에 전용된 SVM을 사용하세요

SVM(스토리지 가상 머신)은 ONTAP 시스템의 테넌트 간에 격리 및 관리적 분리를 제공합니다. SVM을 애플리케이션에 전용하면 권한을 위임하고 리소스 소비를 제한하는 모범 사례를 적용할 수 있습니다.

SVM 관리에는 여러 가지 옵션이 있습니다.

- 백엔드 구성에서 클러스터 관리 인터페이스를 적절한 자격 증명과 함께 제공하고 SVM 이름을 지정합니다.
- ONTAP 시스템 관리자나 CLI를 사용하여 SVM에 대한 전용 관리 인터페이스를 만듭니다.
- NFS 데이터 인터페이스와 관리 역할을 공유합니다.

각각의 경우, 인터페이스는 DNS에 있어야 하며, Trident 구성할 때 DNS 이름을 사용해야 합니다. 이를 통해 네트워크 ID 보존을 사용하지 않고도 SVM-DR과 같은 일부 DR 시나리오를 원활하게 진행할 수 있습니다.

SVM에 대한 전용 관리 LIF와 공유 관리 LIF 중 어느 것을 선호하느냐는 없습니다. 그러나 선택한 접근 방식에 맞게 네트워크 보안 정책이 일치하는지 확인해야 합니다. 그럼에도 불구하고 관리 LIF는 최대 유연성을 촉진하기 위해 DNS를 통해 액세스할 수 있어야 합니다. "SVM-DR" Trident 와 함께 사용할 수 있습니다.

최대 볼륨 수 제한

ONTAP 스토리지 시스템에는 최대 볼륨 수가 있으며, 이는 소프트웨어 버전과 하드웨어 플랫폼에 따라 다릅니다. 참조하다 "NetApp Hardware Universe" 정확한 한도를 확인하려면 해당 플랫폼과 ONTAP 버전을 확인하세요. 볼륨 수가 소진되면 Trident 뿐만 아니라 모든 스토리지 요청에 대한 프로비저닝 작업이 실패합니다.

트라이던트의 `ontap-nas` 그리고 `ontap-san` 드라이버는 생성된 각 Kubernetes 영구 볼륨(PV)에 대해 FlexVolume을 프로비저닝합니다. 그만큼 `ontap-nas-economy` 드라이버는 200개의 PV마다 약 1개의 FlexVolume을 생성합니다(50~300 사이로 구성 가능). 그만큼 `ontap-san-economy` 드라이버는 100개의 PV마다 약 1개의 FlexVolume을 생성합니다(50~200 사이로 구성 가능). Trident 스토리지 시스템의 사용 가능한 모든 볼륨을 소모하지 못하도록 하려면 SVM에 제한을 설정해야 합니다. 명령줄에서 이 작업을 수행할 수 있습니다.

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

에 대한 가치 `max-volumes` 사용자 환경에 따라 여러 가지 기준에 따라 다릅니다.

- ONTAP 클러스터에 있는 기존 볼륨의 수
- 다른 애플리케이션을 위해 Trident 외부에서 프로비저닝할 것으로 예상되는 볼륨 수
- Kubernetes 애플리케이션에서 사용될 것으로 예상되는 영구 볼륨 수

그만큼 `max-volumes` 값은 개별 ONTAP 노드가 아닌 ONTAP 클러스터의 모든 노드에 프로비저닝된 총 볼륨입니다. 결과적으로 ONTAP 클러스터 노드가 다른 노드보다 훨씬 더 많거나 적은 Trident 프로비저닝 볼륨을 갖는 경우가 발생할 수 있습니다.

예를 들어, 2노드 ONTAP 클러스터는 최대 2000개의 FlexVol 볼륨을 호스팅할 수 있습니다. 최대 볼륨 수를 1250으로 설정하는 것은 매우 타당한 것으로 보입니다. 그러나 만약에만 "집계" 한 노드에서 SVM에 할당되거나 한 노드에서 할당된 집계를 프로비저닝할 수 없는 경우(예: 용량 문제로 인해), 다른 노드가 모든 Trident 프로비저닝 볼륨의 대상이 됩니다. 이는 해당 노드의 볼륨 제한에 도달했을 수 있음을 의미합니다. `max-volumes` 값에 도달하면 Trident 와 해당 노드를 사용하는 다른 볼륨 작업에 영향을 미칩니다. 클러스터의 각 노드에서 집계된 데이터가 **Trident** 에서 사용하는 **SVM**에 동일한 개수로 할당되도록 하면 이러한 상황을 피할 수 있습니다**.**

볼륨 복제

NetApp Trident 다음을 사용할 때 볼륨 복제를 지원합니다. `ontap-nas` , `ontap-san` , `solidfire-san` , 그리고 `gcp-cvs` 스토리지 드라이버. 사용 시 `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버, 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새로운 스냅샷이 생성됩니다.

> 다른 StorageClass와 연결된 PVC를 복제하지 마세요. 호환성을 보장하고 예상치 못한 동작을 방지하려면 동일한 StorageClass 내에서 복제 작업을 수행합니다.

Trident 에서 생성된 볼륨의 최대 크기 제한

Trident 에서 생성할 수 있는 볼륨의 최대 크기를 구성하려면 다음을 사용하십시오. `limitVolumeSize` 귀하의 매개변수 `backend.json` 정의.

스토리지 어레이에서 볼륨 크기를 제어하는 것 외에도 Kubernetes 기능도 활용해야 합니다.

Trident 에서 생성된 FlexVol의 최대 크기 제한

ontap-san-economy 및 ontap-nas-economy 드라이버의 풀로 사용되는 FlexVol의 최대 크기를 구성하려면 다음을 사용하세요. `limitVolumePoolSize` 귀하의 매개변수 `backend.json` 정의.

양방향 CHAP를 사용하도록 Trident 구성

백엔드 정의에서 CHAP 초기자 및 대상 사용자 이름과 비밀번호를 지정하고 Trident SVM에서 CHAP를 활성화하도록 할 수 있습니다. 를 사용하여 `useCHAP` 백엔드 구성의 매개변수를 통해 Trident CHAP를 사용하여 ONTAP 백엔드에 대한 iSCSI 연결을 인증합니다.

SVM QoS 정책 생성 및 사용

SVM에 적용된 ONTAP QoS 정책을 활용하면 Trident 프로비저닝 볼륨에서 사용할 수 있는 IOPS 수가 제한됩니다. 이것은 도움이 됩니다 "괴롭힘을 예방하다" 또는 Trident SVM 외부의 작업 부하에 영향을 미치는 제어 불가능한 컨테이너입니다.

몇 단계만으로 SVM에 대한 QoS 정책을 만들 수 있습니다. 가장 정확한 정보는 해당 ONTAP 버전의 설명서를 참조하세요. 아래 예에서는 SVM에서 사용할 수 있는 총 IOPS를 5000으로 제한하는 QoS 정책을 만듭니다.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

또한 ONTAP 버전에서 지원하는 경우 컨테이너화된 워크로드에 대한 처리량을 보장하기 위해 QoS 최소값을 사용하는 것을 고려할 수 있습니다. 적응형 QoS는 SVM 수준 정책과 호환되지 않습니다.

컨테이너화된 워크로드에 전담된 IOPS 수는 여러 측면에 따라 달라집니다. 여기에는 다음이 포함됩니다.

- 스토리지 어레이를 사용하는 다른 워크로드. Kubernetes 배포와 관련이 없는 다른 워크로드가 스토리지 리소스를 활용하는 경우, 해당 워크로드가 실수로 부정적인 영향을 받지 않도록 주의해야 합니다.
- 컨테이너에서 실행되는 예상 작업 부하. 높은 IOPS 요구 사항이 있는 워크로드가 컨테이너에서 실행되는 경우, 낮은 QoS 정책은 나쁜 사용자 경험을 초래합니다.

SVM 수준에서 할당된 QoS 정책은 SVM에 프로비저닝된 모든 볼륨이 동일한 IOPS 풀을 공유하게 된다는 점을 기억하는 것이 중요합니다. 컨테이너화된 애플리케이션 중 하나 또는 소수가 높은 IOPS 요구 사항을 갖는 경우 다른 컨테이너화된 워크로드에 방해가 될 수 있습니다. 이 경우 외부 자동화를 사용하여 볼륨별 QoS 정책을 할당하는 것을 고려해 볼 수 있습니다.

ONTAP 버전이 9.8 이전인 경우에만 QoS 정책 그룹을 SVM에 할당해야 합니다.

Trident 에 대한 **QoS** 정책 그룹 생성

서비스 품질(QoS)은 중요한 작업 부하의 성능이 경쟁 작업 부하로 인해 저하되지 않도록 보장합니다. ONTAP QoS 정책 그룹은 볼륨에 대한 QoS 옵션을 제공하고 사용자가 하나 이상의 작업 부하에 대한 처리량 상한을 정의할 수 있도록 합니다. QoS에 대한 자세한 내용은 다음을 참조하세요. "QoS로 처리량 보장" . 백엔드나 스토리지 풀에서 QoS 정책 그룹을 지정할 수 있으며, 해당 정책 그룹은 해당 풀이나 백엔드에서 생성된 각 볼륨에 적용됩니다.

ONTAP 에는 기존 QoS 정책 그룹과 적응형 QoS 정책 그룹 두 가지가 있습니다. 기존 정책 그룹은 IOPS에서 일정한 최대(또는 이후 버전에서는 최소) 처리량을 제공합니다. 적응형 QoS는 작업 부하 크기에 맞게 처리량을 자동으로 조정하여 작업 부하 크기가 변해도 IOPS 대 TB|GB 비율을 유지합니다. 이는 대규모 배포에서 수백 또는 수천 개의 작업 부하를 관리할 때 상당한 이점을 제공합니다.

QoS 정책 그룹을 생성할 때 다음 사항을 고려하세요.

- 당신은 설정해야합니다 `qosPolicy` 키에 `defaults` 백엔드 구성의 블록. 다음 백엔드 구성 예를 참조하세요.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- 정책 그룹에서 지정한 대로 각 볼륨이 전체 처리량을 얻도록 볼륨별로 정책 그룹을 적용해야 합니다. 공유 정책 그룹은 지원되지 않습니다.

QoS 정책 그룹에 대한 자세한 내용은 다음을 참조하세요. "ONTAP 명령 참조" .

Kubernetes 클러스터 멤버에 대한 스토리지 리소스 액세스 제한

Trident 가 생성한 NFS 볼륨, iSCSI LUN 및 FC LUN에 대한 액세스를 제한하는 것은 Kubernetes 배포에 대한 보안 태세의 중요한 구성 요소입니다. 이렇게 하면 Kubernetes 클러스터에 속하지 않은 호스트가 볼륨에 액세스하여 예기치 않게 데이터를 수정할 가능성이 방지됩니다.

네임스페이스가 Kubernetes 리소스의 논리적 경계라는 것을 이해하는 것이 중요합니다. 동일한 네임스페이스에 있는 리소스는 공유할 수 있다고 가정하지만, 중요한 점은 네임스페이스 간 기능이 없다는 것입니다. 즉, PV는 글로벌 객체이지만 PVC에 바인딩되면 동일한 네임스페이스에 있는 Pod에서만 액세스할 수 있습니다. 적절한 경우 네임스페이스를 사용하여 분리하는 것이 중요합니다.

Kubernetes 컨텍스트에서 데이터 보안과 관련하여 대부분 조직이 가장 우려하는 점은 컨테이너의 프로세스가 호스트에 마운트된 저장소에 액세스할 수 있지만 해당 저장소가 컨테이너용으로 의도되지 않았다는 것입니다. "네임스페이스" 이러한 유형의 손상을 방지하기 위해 설계되었습니다. 하지만 예외가 하나 있습니다. 특권 컨테이너입니다.

특권 컨테이너는 일반적인 컨테이너보다 훨씬 더 많은 호스트 수준 권한으로 실행되는 컨테이너입니다. 이러한 기능은 기본적으로 거부되지 않으므로 다음을 사용하여 기능을 비활성화해야 합니다. "포드 보안 정책" .

Kubernetes와 외부 호스트 모두에서 액세스가 필요한 볼륨의 경우, PV는 관리자가 도입하고 Trident 가 관리하지 않는 기존 방식으로 스토리지를 관리해야 합니다. 이렇게 하면 Kubernetes와 외부 호스트가 모두 연결이 끊기고 더 이상 볼륨을 사용하지 않을 때만 스토리지 볼륨이 파기됩니다. 또한 사용자 정의 내보내기 정책을 적용하면 Kubernetes 클러스터 노드와 Kubernetes 클러스터 외부의 대상 서버에서 액세스할 수 있습니다.

전용 인프라 노드(예: OpenShift)나 사용자 애플리케이션을 예약할 수 없는 다른 노드가 있는 배포의 경우, 별도의 내보내기 정책을 사용하여 스토리지 리소스에 대한 액세스를 추가로 제한해야 합니다. 여기에는 인프라 노드에 배포된 서비스(예: OpenShift Metrics 및 Logging 서비스)와 비인프라 노드에 배포된 표준 애플리케이션에 대한 내보내기 정책을 만드는 작업이 포함됩니다.

전담 수출 정책을 사용하세요

Kubernetes 클러스터에 있는 노드에만 액세스를 허용하는 각 백엔드에 대한 내보내기 정책이 있는지 확인해야 합니다. Trident 자동으로 수출 정책을 만들고 관리할 수 있습니다. 이런 방식으로 Trident Kubernetes 클러스터의 노드에 프로비저닝하는 볼륨에 대한 액세스를 제한하고 노드 추가/삭제를 간소화합니다.

또는 수동으로 내보내기 정책을 만들고 각 노드 액세스 요청을 처리하는 하나 이상의 내보내기 규칙으로 채울 수도 있습니다.

- 사용하다 `vserver export-policy create` ONTAP CLI 명령을 사용하여 내보내기 정책을 생성합니다.
- 다음을 사용하여 내보내기 정책에 규칙을 추가합니다. `vserver export-policy rule create` ONTAP CLI 명령.

이러한 명령을 실행하면 어떤 Kubernetes 노드가 데이터에 액세스할 수 있는지 제한할 수 있습니다.

장애를 입히다 `showmount` **SVM** 응용 프로그램을 위해

그만큼 `showmount` 이 기능을 사용하면 NFS 클라이언트가 SVM에 사용 가능한 NFS 내보내기 목록을 쿼리할 수 있습니다. Kubernetes 클러스터에 배포된 Pod는 다음을 발행할 수 있습니다. `showmount -e` 명령을 내리고, 접근할

수 없는 탈것을 포함하여 사용 가능한 탈것의 목록을 받습니다. 이것 자체는 보안을 위협하는 것은 아니지만 불필요한 정보를 제공하여 권한이 없는 사용자가 NFS 내보내기에 연결하는 데 도움이 될 수 있습니다.

비활성화해야 합니다 `showmount` SVM 수준 ONTAP CLI 명령을 사용하여:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 모범 사례

Trident 위한 SolidFire 스토리지 구성에 대한 모범 사례를 알아보세요.

Solidfire 계정 생성

각 SolidFire 계정은 고유한 볼륨 소유자를 나타내며 고유한 CHAP(Challenge-Handshake 인증 프로토콜) 자격 증명 세트를 받습니다. 계정 이름과 관련 CHAP 자격 증명을 사용하거나 볼륨 액세스 그룹을 통해 계정에 할당된 볼륨에 액세스할 수 있습니다. 한 계정에는 최대 2,000개의 볼륨이 할당될 수 있지만, 볼륨은 한 계정에만 속할 수 있습니다.

QoS 정책 생성

여러 볼륨에 적용할 수 있는 표준화된 서비스 품질 설정을 만들고 저장하려면 SolidFire 서비스 품질(QoS) 정책을 사용하세요.

볼륨별로 QoS 매개변수를 설정할 수 있습니다. 각 볼륨의 성능은 QoS를 정의하는 세 가지 구성 가능한 매개변수(최소 IOPS, 최대 IOPS, 버스트 IOPS)를 설정하여 보장할 수 있습니다.

4Kb 블록 크기에 대한 최소, 최대 및 버스트 IOPS 값은 다음과 같습니다.

IOPS 매개변수	정의	최소값	기본값	최대값(4Kb)
최소 IOPS	볼륨에 대해 보장된 성능 수준입니다.	50	50	15000
최대 IOPS	성능은 이 한도를 초과하지 않습니다.	50	15000	200,000
버스트 IOPS	짧은 버스트 시나리오에서 허용되는 최대 IOPS입니다.	50	15000	200,000

최대 IOPS와 버스트 IOPS는 최대 200,000까지 설정할 수 있지만, 볼륨의 실제 최대 성능은 클러스터 사용량과 노드당 성능에 따라 제한됩니다.

블록 크기와 대역폭은 IOPS 수에 직접적인 영향을 미칩니다. 블록 크기가 증가함에 따라 시스템은 더 큰 블록 크기를 처리하는 데 필요한 수준으로 대역폭을 늘립니다. 대역폭이 증가함에 따라 시스템이 달성할 수 있는 IOPS 수는 감소합니다. 참조하다 "SolidFire 서비스 품질" QoS 및 성능에 대한 자세한 내용은 다음을 참조하세요.

SolidFire 인증

Element는 CHAP 및 VAG(볼륨 액세스 그룹)라는 두 가지 인증 방법을 지원합니다. CHAP는 CHAP 프로토콜을 사용하여 백엔드에 호스트를 인증합니다. 볼륨 액세스 그룹은 프로비저닝하는 볼륨에 대한 액세스를 제어합니다. NetApp 인증에 CHAP를 사용할 것을 권장합니다. CHAP는 더 간단하고 확장 제한이 없기 때문입니다.

> 향상된 CSI 프로비저너가 탑재된 Trident CHAP 인증 사용을 지원합니다. VAG는 기존의 비 CSI 작동 모드에서만 사용해야 합니다.

CHAP 인증(개시자가 의도된 볼륨 사용자인지 확인하는 것)은 계정 기반 액세스 제어를 통해서만 지원됩니다. 인증에 CHAP를 사용하는 경우 단방향 CHAP와 양방향 CHAP의 두 가지 옵션을 사용할 수 있습니다. 단방향 CHAP는 SolidFire 계정 이름과 개시자 비밀번호를 사용하여 볼륨 액세스를 인증합니다. 양방향 CHAP 옵션은 볼륨이 계정 이름과 개시자 비밀번호를 통해 호스트를 인증하고, 그런 다음 호스트가 계정 이름과 대상 비밀번호를 통해 볼륨을 인증하기 때문에 볼륨을 인증하는 가장 안전한 방법을 제공합니다.

하지만 CHAP를 활성화할 수 없고 VAG가 필요한 경우 액세스 그룹을 만들고 호스트 이니시에이터와 볼륨을 액세스 그룹에 추가합니다. 액세스 그룹에 추가하는 각 IQN은 CHAP 인증 여부와 관계없이 그룹의 각 볼륨에 액세스할 수 있습니다. iSCSI 이니시에이터가 CHAP 인증을 사용하도록 구성된 경우 계정 기반 액세스 제어가 사용됩니다. iSCSI 이니시에이터가 CHAP 인증을 사용하도록 구성되지 않은 경우 볼륨 액세스 그룹 액세스 제어가 사용됩니다.

더 많은 정보는 어디에서 찾을 수 있나요?

다음은 모범 사례 문서 중 일부입니다. 검색 "NetApp 라이브러리" 최신 버전을 보려면 여기를 클릭하세요.

- ONTAP*
- "NFS 모범 사례 및 구현 가이드"
- "SAN 관리"(iSCSI용)
- "RHEL용 iSCSI Express 구성"

엘리먼트 소프트웨어

- "Linux용 SolidFire 구성"
- NetApp HCI*
- "NetApp HCI 배포 전제 조건"
- "NetApp 배포 엔진에 액세스"

응용 프로그램 모범 사례 정보

- "ONTAP 에서 MySQL을 위한 모범 사례"
- "SolidFire 에서 MySQL을 위한 모범 사례"
- "NetApp SolidFire 및 Cassandra"
- "SolidFire 에 대한 Oracle 모범 사례"
- "SolidFire 에서의 PostgreSQL 모범 사례"

모든 애플리케이션에 특정 지침이 있는 것은 아니므로 NetApp 팀과 협력하여 사용하는 것이 중요합니다. "NetApp 라이브러리" 가장 최신 문서를 찾으려면.

Trident 통합

Trident 통합하려면 드라이버 선택 및 배포, 스토리지 클래스 설계, 가상 풀 설계, 스토리지 프로비저닝에 미치는 PVC(영구 볼륨 클레임)의 영향, 볼륨 작업, Trident 사용한 OpenShift 서비스 배포 등의 설계 및 아키텍처 요소를 통합해야 합니다.

드라이버 선택 및 배치

스토리지 시스템에 맞는 백엔드 드라이버를 선택하고 배포합니다.

ONTAP 백엔드 드라이버

ONTAP 백엔드 드라이버는 사용되는 프로토콜과 스토리지 시스템에서 볼륨이 프로비저닝되는 방식에 따라 구분됩니다. 따라서 어떤 드라이버를 배치할지 결정할 때 신중하게 고려하세요.

더 높은 수준에서, 애플리케이션에 공유 스토리지(동일한 PVC에 액세스하는 여러 포드)가 필요한 구성 요소가 있는 경우 NAS 기반 드라이버가 기본 선택이 되고, 블록 기반 iSCSI 드라이버는 공유되지 않는 스토리지의 요구 사항을 충족합니다. 애플리케이션의 요구 사항과 스토리지 및 인프라 팀의 편의성 수준에 따라 프로토콜을 선택하세요. 일반적으로 대부분의 애플리케이션에서는 두 가지 사이에 큰 차이가 없으므로, 공유 스토리지(두 개 이상의 포드에 동시에 액세스해야 하는 경우)가 필요한지 여부에 따라 결정이 내려지는 경우가 많습니다.

사용 가능한 ONTAP 백엔드 드라이버는 다음과 같습니다.

- `ontap-nas`: 프로비저닝된 각 PV는 전체 ONTAP FlexVolume입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 qtree이며, FlexVolume당 구성 가능한 qtree 수가 있습니다 (기본값은 200).
- `ontap-nas-flexgroup`: 각 PV는 전체 ONTAP FlexGroup 으로 프로비저닝되고, SVM에 할당된 모든 집계가 사용됩니다.
- `ontap-san`: 프로비저닝된 각 PV는 자체 FlexVolume 내의 LUN입니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 LUN이며, FlexVolume당 LUN 수는 구성 가능합니다(기본값은 100).

세 가지 NAS 드라이버 중에서 선택하면 애플리케이션에서 사용할 수 있는 기능에 몇 가지 영향이 있습니다.

아래 표에서 모든 기능이 Trident 통해 공개되는 것은 아니라는 점에 유의하세요. 해당 기능이 필요한 경우 스토리지 관리자가 프로비저닝 후에 일부 기능을 적용해야 합니다. 상위 첨자 각주는 기능과 드라이버별 기능을 구분합니다.

ONTAP NAS 드라이버	스냅샷	클론	동적 수출 정책	다중 부착	서비스 품질	크기 조정	복제
`ontap-nas`	예	예	예각주:5[]	예	예각주:1[]	예	예각주:1[]
`ontap-nas-economy`	NO각주:3[]	NO각주:3[]	예각주:5[]	예	NO각주:3[]	예	NO각주:3[]
`ontap-nas-flexgroup`	예각주:1[]	아니요	예각주:5[]	예	예각주:1[]	예	예각주:1[]

Trident ONTAP 용 SAN 드라이버 2개를 제공하며, 해당 드라이버의 기능은 아래와 같습니다.

ONTAP SAN 드라이버	스냅샷	클론	다중 부착	양방향 **CHAP**	서비스 품질	크기 조정	복제
`ontap-san`	예	예	예각주:4[]	예	예각주:1[]	예	예각주:1[]
`ontap-san-economy`	예	예	예각주:4[]	예	NO각주:3[]	예	NO각주:3[]

> 위 표에 대한 각주: 예각주:1[]: Trident 에서 관리하지 않음 예각주:2[]: Trident 에서 관리하지만 PV 세분화되지 않음 아니요각주:3[]: Trident 에서 관리하지 않고 PV 세분화되지 않음 예각주:4[]: 원시 블록 볼륨에 대해 지원됨 예각주:5[]: Trident 에서 지원됨

PV 세분화가 아닌 기능은 전체 FlexVolume에 적용되며 모든 PV(즉, 공유 FlexVol의 qtree 또는 LUN)는 공통 일정을 공유합니다.

위의 표에서 볼 수 있듯이, 다음 기능 간의 많은 부분이 `ontap-nas` 그리고 `ontap-nas-economy` 동일합니다. 그러나, `ontap-nas-economy` 드라이버는 PV 단위로 일정을 제어하는 기능을 제한하며, 이는 특히 재해 복구 및 백업 계획에 영향을 미칠 수 있습니다. ONTAP 스토리지에서 PVC 복제 기능을 활용하려는 개발 팀의 경우 이는 다음을 사용할 때만 가능합니다. `ontap-nas` , `ontap-san` 또는 `ontap-san-economy` 운전자.

> 그만큼 `solidfire-san` 드라이버는 PVC를 복제할 수도 있습니다.

Cloud Volumes ONTAP 백엔드 드라이버

Cloud Volumes ONTAP 파일 공유 및 NAS 및 SAN 프로토콜(NFS, SMB/CIFS, iSCSI)을 제공하는 블록 수준 스토리지를 포함하여 다양한 사용 사례에 대한 엔터프라이즈급 스토리지 기능과 함께 데이터 제어 기능을 제공합니다. Cloud Volume ONTAP 에 호환되는 드라이버는 다음과 같습니다. `ontap-nas` , `ontap-nas-economy` , `ontap-san` 그리고 `ontap-san-economy` . 이러한 기능은 Azure용 Cloud Volume ONTAP , GCP용 Cloud Volume ONTAP 에 적용됩니다.

Amazon FSx for ONTAP 백엔드 드라이버

Amazon FSx for NetApp ONTAP 사용하면 AWS에 데이터를 저장함으로써 얻는 단순성, 민첩성, 보안 및 확장성의 이점을 누리면서 익숙한 NetApp 기능, 성능 및 관리 역량을 활용할 수 있습니다. FSx for ONTAP 다양한 ONTAP 파일 시스템 기능과 관리 API를 지원합니다. Cloud Volume ONTAP 에 호환되는 드라이버는 다음과 같습니다. `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , `ontap-san` 그리고 `ontap-san-economy` .

NetApp HCI/ SolidFire 백엔드 드라이버

그만큼 `solidfire-san` NetApp HCI/ SolidFire 플랫폼과 함께 사용되는 드라이버는 관리자가 QoS 제한에 따라 Trident 에 대한 Element 백엔드를 구성하는 데 도움이 됩니다. Trident 에서 프로비저닝한 볼륨에 대한 특정 QoS 제한을 설정하도록 백엔드를 설계하려면 다음을 사용하십시오. `type` 백엔드 파일의 매개변수. 관리자는 또한 다음을 사용하여 저장소에서 생성될 수 있는 볼륨 크기를 제한할 수 있습니다. `limitVolumeSize` 매개변수. 현재 볼륨 크기 조정 및 볼륨 복제와 같은 Element 스토리지 기능은 지원되지 않습니다. `solidfire-san` 운전사. 이러한 작업은 Element Software 웹 UI를 통해 수동으로 수행해야 합니다.

SolidFire 드라이버	스냅샷	클론	다중 부착	녀석	서비스 품질	크기 조정	복제
`solidfire-san`	예	예	예각주:2[]	예	예	예	예각주:1[]

> 각주: 예각주:1[]: Trident 에서 관리되지 않음 예각주:2[]: 원시 블록 볼륨에 대해 지원됨

Azure NetApp Files 백엔드 드라이버

Trident 다음을 사용합니다. `azure-netapp-files` 관리하는 드라이버"Azure NetApp Files" 서비스.

이 드라이버에 대한 자세한 정보와 이를 구성하는 방법은 다음에서 확인할 수 있습니다."Azure NetApp Files 대한 Trident 백엔드 구성" .

Azure NetApp Files 드라이버	스냅샷	클론	다중 부착	서비스 품질	확장하다	복제
`azure-netapp-files`	예	예	예	예	예	예각주:1[]

각주: 예각주:1[]: Trident 에서 관리하지 않음

Google Cloud 백엔드 드라이버의 Cloud Volumes Service

Trident 다음을 사용합니다. `gcp-cvs` Google Cloud의 Cloud Volumes Service 에 연결하는 드라이버입니다.

그만큼 `gcp-cvs` 드라이버는 가상 풀을 사용하여 백엔드를 추상화하고 Trident 볼륨 배치를 결정할 수 있도록 합니다. 관리자는 가상 풀을 정의합니다. `backend.json` 파일. 저장소 클래스는 선택기를 사용하여 레이블로 가상 풀을 식별합니다.

- 백엔드에 가상 풀이 정의된 경우 Trident 해당 가상 풀이 제한된 Google Cloud 스토리지 풀에 볼륨을 생성하려고 시도합니다.
- 백엔드에 가상 풀이 정의되어 있지 않으면 Trident 해당 지역의 사용 가능한 스토리지 풀 중에서 Google Cloud 스토리지 풀을 선택합니다.

Trident 에서 Google Cloud 백엔드를 구성하려면 다음을 지정해야 합니다. `projectNumber` , `apiRegion` , 그리고 `apiKey` 백엔드 파일에서. 프로젝트 번호는 Google Cloud 콘솔에서 찾을 수 있습니다. API 키는 Google Cloud에서 Cloud Volumes Service 에 대한 API 액세스를 설정할 때 생성한 서비스 계정 개인 키 파일에서 가져옵니다.

Google Cloud 서비스 유형 및 서비스 수준의 Cloud Volumes Service 에 대한 자세한 내용은 다음을 참조하세요 ."GCP용 CVS에 대한 Trident 지원에 대해 알아보세요" .

Google Cloud 드라이버용 **Cloud Volumes Service**	스냅샷	클론	다중 부착	서비스 품질	확장하다	복제
`gcp-cvs`	예	예	예	예	예	CVS-Performance 서비스 유형에서만 사용 가능합니다.

복제 노트

- 복제는 Trident 에서 관리되지 않습니다.
- 복제본은 소스 볼륨과 동일한 스토리지 풀에 생성됩니다.

스토리지 클래스 디자인

Kubernetes 스토리지 클래스 객체를 생성하려면 개별 스토리지 클래스를 구성하고 적용해야 합니다. 이 섹션에서는 애플리케이션에 맞는 스토리지 클래스를 설계하는 방법을 설명합니다.

특정 백엔드 활용

필터링은 특정 스토리지 클래스 개체 내에서 사용하여 해당 특정 스토리지 클래스와 함께 사용할 스토리지 풀 또는 풀 세트를 결정할 수 있습니다. 스토리지 클래스에는 세 가지 필터 세트를 설정할 수 있습니다. `storagePools` , `additionalStoragePools` , 및/또는 `excludeStoragePools` .

그만큼 `storagePools` 매개변수는 지정된 속성과 일치하는 풀 세트로 저장소를 제한하는 데 도움이 됩니다. 그만큼 `additionalStoragePools` 매개변수는 Trident 가 프로비저닝에 사용하는 풀 세트와 속성에 의해 선택된 풀 세트를 확장하는 데 사용됩니다. `storagePools` 매개변수. 두 매개변수 중 하나만 사용하거나 두 매개변수를 함께 사용하여 적절한 스토리지 풀 세트가 선택되었는지 확인할 수 있습니다.

그만큼 `excludeStoragePools` 매개변수는 속성과 일치하는 풀의 나열된 세트를 구체적으로 제외하는 데 사용됩니다.

QoS 정책 에뮬레이션

서비스 품질 정책을 에뮬레이트하기 위해 스토리지 클래스를 설계하려면 다음을 사용하여 스토리지 클래스를 만듭니다. `media` 속성으로 `hdd` 또는 `ssd` . 에 근거하여 `media` 저장 클래스에 언급된 속성에 대해 Trident 적절한 백엔드를 선택합니다. `hdd` 또는 `ssd` 미디어 속성과 일치하는 집계를 수행한 다음 볼륨 프로비저닝을 특정 집계로 지시합니다. 따라서 PREMIUM 스토리지 클래스를 생성할 수 있습니다. `media` 속성 집합으로 `ssd` 이는 PREMIUM QoS 정책으로 분류될 수 있습니다. 또 다른 저장 클래스 STANDARD를 만들면 미디어 속성이 `hdd`로 설정되고, 이는 STANDARD QoS 정책으로 분류될 수 있습니다. 저장소 클래스에서 ``IOPS'' 속성을 사용하여 프로비저닝을 QoS 정책으로 정의할 수 있는 Element 어플라이언스로 리디렉션할 수도 있습니다.

특정 기능에 기반한 백엔드 활용

스토리지 클래스는 씬 및 씩 프로비저닝, 스냅샷, 복제, 암호화 등의 기능이 활성화된 특정 백엔드에서 볼륨 프로비저닝을 지시하도록 설계될 수 있습니다. 사용할 저장소를 지정하려면 필요한 기능이 활성화된 적절한 백엔드를 지정하는 저장소 클래스를 만듭니다.

가상 풀

모든 Trident 백엔드에서 가상 풀을 사용할 수 있습니다. Trident 제공하는 모든 드라이버를 사용하여 모든 백엔드에 대한 가상 풀을 정의할 수 있습니다.

가상 풀을 사용하면 관리자가 백엔드에 대한 추상화 수준을 생성할 수 있으며, 이는 스토리지 클래스를 통해 참조할 수 있어 백엔드에 볼륨을 보다 유연하게 배치하고 효율적으로 배치할 수 있습니다. 동일한 서비스 클래스로 서로 다른 백엔드를 정의할 수 있습니다. 게다가 동일한 백엔드에 서로 다른 특성을 가진 여러 개의 스토리지 풀을 만들 수도 있습니다. 스토리지 클래스가 특정 레이블이 있는 선택기로 구성된 경우 Trident 모든 선택기 레이블과 일치하는 백엔드를 선택하여 볼륨을 배치합니다. 스토리지 클래스 선택기 레이블이 여러 스토리지 풀과 일치하는 경우 Trident 볼륨을 프로비저닝할 스토리지 풀을 하나 선택합니다.

가상 풀 디자인

백엔드를 생성할 때 일반적으로 매개변수 집합을 지정할 수 있습니다. 이전에는 관리자가 동일한 스토리지 자격 증명과 다른 매개변수 집합을 사용하여 다른 백엔드를 생성하는 것이 불가능했습니다. 가상 풀이 도입되면서 이 문제가 해결되었습니다. 가상 풀은 백엔드와 Kubernetes 스토리지 클래스 사이에 도입된 레벨 추상화로, 관리자가 백엔드에

관계없이 Kubernetes 스토리지 클래스를 통해 선택기로 참조할 수 있는 레이블과 함께 매개변수를 정의할 수 있도록 합니다. 가상 풀은 Trident 를 통해 지원되는 모든 NetApp 백엔드에 대해 정의할 수 있습니다. 해당 목록에는 SolidFire/NetApp HCI, ONTAP, GCP의 Cloud Volumes Service 및 Azure NetApp Files 포함됩니다.

> 가상 풀을 정의할 때 백엔드 정의에서 기존 가상 풀의 순서를 재정렬하지 않는 것이 좋습니다. 기존 가상 풀의 속성을 편집/수정하지 않고 대신 새 가상 풀을 정의하는 것이 좋습니다.

다양한 서비스 수준/**QoS** 에뮬레이션

서비스 클래스를 에뮬레이션하기 위해 가상 풀을 설계하는 것이 가능합니다. Azure NetApp Files 용 Cloud Volume Service의 가상 풀 구현을 사용하여 다양한 서비스 클래스를 설정하는 방법을 살펴보겠습니다. 다양한 성능 수준을 나타내는 여러 레이블로 Azure NetApp Files 백엔드를 구성합니다. 세트 `servicelevel` 각 라벨 아래에 필요한 다른 측면을 추가하고, 적절한 성능 수준에 맞춰 측면을 조정합니다. 이제 다양한 가상 풀에 매핑되는 다양한 Kubernetes 스토리지 클래스를 만듭니다. 를 사용하여 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다.

특정 측면 집합 할당

특정 측면을 갖춘 여러 개의 가상 풀을 단일 스토리지 백엔드에서 설계할 수 있습니다. 이를 위해 백엔드를 여러 개의 레이블로 구성하고 각 레이블 아래에 필요한 측면을 설정합니다. 이제 다음을 사용하여 다양한 Kubernetes 스토리지 클래스를 만듭니다. `parameters.selector` 다양한 가상 풀에 매핑되는 필드입니다. 백엔드에 프로비저닝되는 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

저장 용량에 영향을 미치는 **PVC** 특성

요청된 스토리지 클래스를 넘어서는 일부 매개변수는 PVC를 생성할 때 Trident 프로비저닝 결정 프로세스에 영향을 미칠 수 있습니다.

접근 모드

PVC를 통해 저장소를 요청할 때 필수 필드 중 하나는 액세스 모드입니다. 원하는 모드는 저장 요청을 호스팅하기 위해 선택된 백엔드에 영향을 미칠 수 있습니다.

Trident 다음 매트릭스에 따라 지정된 액세스 방법과 사용된 저장 프로토콜을 일치시키려고 시도합니다. 이는 기본 저장 플랫폼과 무관합니다.

	한 번 읽기/쓰기	읽기 전용 다수	읽기쓰기많음
iSCSI	예	예	네 (원시 블록)
NFS	예	예	예

NFS 백엔드가 구성되지 않은 Trident 배포에 ReadWriteMany PVC에 대한 요청을 제출하면 볼륨이 프로비저닝되지 않습니다. 이러한 이유로 요청자는 자신의 애플리케이션에 적합한 액세스 모드를 사용해야 합니다.

볼륨 작업

영구 볼륨 수정

두 가지 예외를 제외하고 영구 볼륨은 Kubernetes에서 변경할 수 없는 객체입니다. 회수 정책과 크기를 생성한 후에는 이를 수정할 수 있습니다. 하지만 이렇게 해도 볼륨의 일부 측면이 Kubernetes 외부에서 수정되는 것을 막을 수는 없습니다. 이는 특정 애플리케이션에 맞게 볼륨을 사용자 지정하거나, 용량이 실수로 소모되는 것을 방지하거나, 어떤

이유로든 볼륨을 다른 스토리지 컨트롤러로 옮기는 경우에 바람직할 수 있습니다.

> Kubernetes 인트리 프로비저너는 현재 NFS, iSCSI 또는 FC PV에 대한 볼륨 크기 조정 작업을 지원하지 않습니다. Trident NFS, iSCSI, FC 볼륨 확장을 모두 지원합니다.

PV의 연결 세부정보는 생성 후 수정할 수 없습니다.

주문형 볼륨 스냅샷 만들기

Trident CSI 프레임워크를 사용하여 주문형 볼륨 스냅샷 생성과 스냅샷에서 PVC 생성을 지원합니다. 스냅샷은 데이터의 특정 시점 사본을 유지 관리하는 편리한 방법을 제공하며 Kubernetes의 소스 PV와 독립적인 수명 주기를 갖습니다. 이러한 스냅샷은 PVC를 복제하는 데 사용할 수 있습니다.

스냅샷에서 볼륨 생성

Trident 볼륨 스냅샷에서 PersistentVolume을 생성하는 기능도 지원합니다. 이를 달성하려면 PersistentVolumeClaim을 생성하고 다음을 언급하기만 하면 됩니다. `datasource` 볼륨을 생성하는 데 필요한 스냅샷입니다. Trident 스냅샷에 있는 데이터로 볼륨을 생성하여 이 PVC를 처리합니다. 이 기능을 사용하면 여러 지역에 걸쳐 데이터를 복제하고, 테스트 환경을 만들고, 손상되거나 훼손된 운영 볼륨을 전체적으로 교체하거나, 특정 파일과 디렉터리를 검색하여 다른 연결된 볼륨으로 전송할 수 있습니다.

클러스터에서 볼륨 이동

스토리지 관리자는 ONTAP 클러스터의 집계와 컨트롤러 간에 볼륨을 스토리지 소비자에게 중단 없이 이동할 수 있습니다. 대상 집계가 Trident 사용하는 SVM에서 액세스할 수 있는 집계인 한, 이 작업은 Trident 나 Kubernetes 클러스터에 영향을 미치지 않습니다. 중요한 점은, 집계가 SVM에 새로 추가된 경우 백엔드를 Trident 에 다시 추가하여 새로 고쳐야 한다는 것입니다. 이렇게 하면 Trident SVM을 재인벤토리하여 새로운 집계가 인식되도록 합니다.

하지만 백엔드 간 볼륨 이동은 Trident 에서 자동으로 지원되지 않습니다. 여기에는 동일한 클러스터 내의 SVM 간, 클러스터 간 또는 다른 스토리지 플랫폼(해당 스토리지 시스템이 Trident 에 연결된 경우에도 해당)이 포함됩니다.

볼륨을 다른 위치로 복사하는 경우 볼륨 가져오기 기능을 사용하여 현재 볼륨을 Trident 로 가져올 수 있습니다.

볼륨 확장

Trident NFS, iSCSI, FC PV의 크기 조정을 지원합니다. 이를 통해 사용자는 Kubernetes 계층을 통해 볼륨 크기를 직접 조정할 수 있습니다. ONTAP, SolidFire/ NetApp HCI 및 Cloud Volumes Service 백엔드를 포함한 모든 주요 NetApp 스토리지 플랫폼에서 볼륨 확장이 가능합니다. 나중에 확장이 가능하도록 설정하세요. `allowVolumeExpansion` 에게 `true` 볼륨과 연결된 StorageClass에서. 영구 볼륨의 크기를 조정해야 할 때마다 다음을 편집합니다. `spec.resources.requests.storage` 영구 볼륨 클레임에 필요한 볼륨 크기에 대한 주석을 추가합니다. Trident 스토리지 클러스터의 볼륨 크기를 자동으로 조정합니다.

기존 볼륨을 **Kubernetes**로 가져오기

볼륨 가져오기는 기존 스토리지 볼륨을 Kubernetes 환경으로 가져오는 기능을 제공합니다. 이는 현재 다음에서 지원됩니다. `ontap-nas` , `ontap-nas-flexgroup` , `solidfire-san` , `azure-netapp-files` , 그리고 `gcp-cvs` 운전자. 이 기능은 기존 애플리케이션을 Kubernetes로 이식할 때나 재해 복구 시나리오에서 유용합니다.

ONTAP 사용할 때 `solidfire-san` 드라이버는 명령을 사용하세요 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` Trident 에서 관리할 수 있도록 기존 볼륨을 Kubernetes로 가져옵니다. import volume 명령에 사용된 PVC YAML 또는 JSON 파일은 Trident 프로비저너로 식별하는 스토리지 클래스를 가리킵니다. NetApp HCI/ SolidFire 백엔드를 사용하는 경우 볼륨 이름이 고유한지

확인하세요. 볼륨 이름이 중복된 경우 볼륨을 고유한 이름으로 복제하여 볼륨 가져오기 기능에서 구별할 수 있도록 합니다.

만약 `azure-netapp-files` 또는 `gcp-cvs` 드라이버가 사용되면 명령을 사용하세요 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 볼륨을 Kubernetes로 가져와서 Trident에서 관리하도록 합니다. 이를 통해 고유한 볼륨 참조가 보장됩니다.

위 명령을 실행하면 Trident 백엔드에서 볼륨을 찾아 크기를 읽습니다. 구성된 PVC의 볼륨 크기를 자동으로 추가하고 필요한 경우 덮어씁니다. 그런 다음 Trident 새로운 PV를 만들고 Kubernetes가 PVC를 PV에 바인딩합니다.

컨테이너가 특정 수입 PVC를 필요로 하도록 배치된 경우 PVC/PV 쌍이 볼륨 수입 프로세스를 통해 바인딩될 때까지 보류 상태로 유지됩니다. PVC/PV 쌍이 결합된 후 다른 문제가 없다면 컨테이너가 올라와야 합니다.

등록 서비스

레지스트리에 대한 저장소 배포 및 관리가 문서화되었습니다."넷앱.io" 에서"블로그" .

로깅 서비스

다른 OpenShift 서비스와 마찬가지로 로깅 서비스는 플레이북에 제공된 인벤토리 파일(호스트)에서 제공하는 구성 매개변수를 사용하여 Ansible을 사용하여 배포됩니다. 두 가지 설치 방법에 대해 설명합니다. OpenShift를 처음 설치할 때 로깅을 배포하는 방법과 OpenShift를 설치한 후에 로깅을 배포하는 방법입니다.

> Red Hat OpenShift 버전 3.9부터 공식 문서에서는 데이터 손상에 대한 우려로 인해 로깅 서비스에 NFS를 사용하지 않는 것이 좋습니다. 이는 Red Hat에서 자사 제품을 테스트한 결과를 기반으로 합니다. ONTAP NFS 서버는 이러한 문제가 없으며 로깅 배포를 쉽게 지원할 수 있습니다. 궁극적으로 로깅 서비스에 대한 프로토콜을 선택하는 것은 사용자의 몫입니다. NetApp 플랫폼을 사용할 경우 두 프로토콜 모두 잘 작동하며, NFS를 선호한다면 NFS를 피할 이유가 없습니다.

로깅 서비스와 함께 NFS를 사용하도록 선택하는 경우 Ansible 변수를 설정해야 합니다. `openshift_enable_unsupported_configurations` 에게 `true` 설치 프로그램이 실패하는 것을 방지합니다.

시작하기

로깅 서비스는 선택적으로 애플리케이션과 OpenShift 클러스터 자체의 핵심 작업에 모두 배포될 수 있습니다. 변수를 지정하여 작업 로깅을 배포하도록 선택하는 경우 `openshift_logging_use_ops` ~처럼 `true` , 서비스의 인스턴스가 두 개 생성됩니다. 작업에 대한 로깅 인스턴스를 제어하는 변수에는 "ops"가 포함되지만, 애플리케이션에 대한 인스턴스에는 포함되지 않습니다.

기본 서비스에서 올바른 스토리지를 활용하려면 배포 방법에 따라 Ansible 변수를 구성하는 것이 중요합니다. 각 배포 방법에 대한 옵션을 살펴보겠습니다.

> 아래 표에는 로깅 서비스와 관련된 저장소 구성에 관련된 변수만 포함되어 있습니다. 다른 옵션은 다음에서 찾을 수 있습니다."Red Hat OpenShift 로깅 문서" 배포에 맞게 검토, 구성 및 사용해야 합니다.

아래 표의 변수를 사용하면 Ansible 플레이북이 제공된 세부 정보를 사용하여 로깅 서비스에 대한 PV 및 PVC를 생성합니다. 이 방법은 OpenShift를 설치한 후 구성 요소 설치 플레이북을 사용하는 것보다 유연성이 상당히 떨어지지만, 기존 볼륨을 사용할 수 있는 경우에는 한 가지 옵션입니다.

변하기 쉬운	세부
`openshift_logging_storage_kind`	로 설정 `nfs` 설치 프로그램이 로깅 서비스에 대한 NFS PV를 생성하도록 합니다.
`openshift_logging_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이는 가상 머신의 dataLIF로 설정되어야 합니다.
`openshift_logging_storage_nfs_directory`	NFS 내보내기에 대한 마운트 경로입니다. 예를 들어, 볼륨이 다음과 같이 접합된 경우 `/openshift_logging` , 이 변수에 대해 해당 경로를 사용하게 됩니다.
`openshift_logging_storage_volume_name`	이름, 예를 들어 `pv_ose_logs` PV를 생성하려면.
`openshift_logging_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi` .

OpenShift 클러스터가 이미 실행 중이고 Trident 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변하기 쉬운	세부
`openshift_logging_es_pvc_dynamic`	동적으로 프로비저닝된 볼륨을 사용하려면 true로 설정합니다.
`openshift_logging_es_pvc_storage_class_name`	PVC에서 사용될 스토리지 클래스의 이름입니다.
`openshift_logging_es_pvc_size`	PVC에서 요청된 볼륨의 크기입니다.
`openshift_logging_es_pvc_prefix`	로깅 서비스에서 사용하는 PVC에 대한 접두사입니다.
`openshift_logging_es_ops_pvc_dynamic`	로 설정 `true` ops 로깅 인스턴스에 동적으로 프로비저닝된 볼륨을 사용합니다.
`openshift_logging_es_ops_pvc_storage_class_name`	ops 로깅 인스턴스의 스토리지 클래스 이름입니다.
`openshift_logging_es_ops_pvc_size`	ops 인스턴스에 대한 볼륨 요청의 크기입니다.
`openshift_logging_es_ops_pvc_prefix`	ops 인스턴스 PVC에 대한 접두사입니다.

로깅 스택 배포

초기 OpenShift 설치 프로세스의 일부로 로깅을 배포하는 경우 표준 배포 프로세스만 따르면 됩니다. Ansible은 필요한 서비스와 OpenShift 객체를 구성하고 배포하므로 Ansible이 완료되는 즉시 서비스를 사용할 수 있습니다.

하지만 초기 설치 후에 배포하는 경우 Ansible에서 구성 요소 플레이북을 사용해야 합니다. 이 프로세스는 OpenShift의 다른 버전에 따라 약간씩 변경될 수 있으므로 반드시 읽고 따르십시오."Red Hat OpenShift Container Platform 3.11 설명서" 귀하의 버전에 맞게.

메트릭 서비스

메트릭 서비스는 관리자에게 OpenShift 클러스터의 상태, 리소스 활용도, 가용성에 대한 귀중한 정보를 제공합니다. 또한 포드 자동 확장 기능에도 필요하며, 많은 조직에서 요금 청구 및/또는 쇼백 애플리케이션에 메트릭 서비스의 데이터를 사용합니다.

로깅 서비스와 OpenShift 전체와 마찬가지로 Ansible은 메트릭 서비스를 배포하는 데 사용됩니다. 또한 로깅 서비스와 마찬가지로 메트릭 서비스는 클러스터의 초기 설정 중이나 구성 요소 설치 방법을 사용하여 운영이 완료된 후에 배포할 수 있습니다. 다음 표에는 메트릭 서비스에 대한 영구 저장소를 구성할 때 중요한 변수가 포함되어 있습니다.

아래 표에는 메트릭 서비스와 관련된 스토리지 구성에 관련된 변수만 포함되어 있습니다. 배포에 맞게 검토, 구성 및 사용해야 하는 다른 옵션도 설명서에 많이 나와 있습니다.

변하기 쉬운	세부
`openshift_metrics_storage_kind`	로 설정 `nfs` 설치 프로그램이 로깅 서비스에 대한 NFS PV를 생성하도록 합니다.
`openshift_metrics_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이는 SVM의 dataLIF로 설정되어야 합니다.
`openshift_metrics_storage_nfs_directory`	NFS 내보내기에 대한 마운트 경로입니다. 예를 들어, 볼륨이 다음과 같이 접합된 경우 `/openshift_metrics` , 이 변수에 대해 해당 경로를 사용하게 됩니다.
`openshift_metrics_storage_volume_name`	이름, 예를 들어 `pv_ose_metrics` PV를 생성하려면.
`openshift_metrics_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi` .

OpenShift 클러스터가 이미 실행 중이고 Trident 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변하기 쉬운	세부
`openshift_metrics_cassandra_pvc_prefix`	PVC 지표에 사용할 접두사입니다.
`openshift_metrics_cassandra_pvc_size`	요청할 볼륨의 크기입니다.
`openshift_metrics_cassandra_storage_type`	메트릭에 사용할 저장소 유형입니다. Ansible이 적절한 저장소 클래스로 PVC를 생성하려면 이 값을 동적으로 설정해야 합니다.
`openshift_metrics_cassanda_pvc_storage_class_name`	사용할 저장 클래스의 이름입니다.

메트릭 서비스 배포

hosts/inventory 파일에 적절한 Ansible 변수를 정의한 후 Ansible을 사용하여 서비스를 배포합니다. OpenShift 설치 시점에 배포하는 경우 PV가 자동으로 생성되어 사용됩니다. 구성 요소 플레이북을 사용하여 배포하는 경우 OpenShift를 설치한 후 Ansible이 필요한 PVC를 생성하고 Trident 해당 PVC에 대한 스토리지를 프로비저닝한 후 서비스를 배포합니다.

위의 변수와 배포 프로세스는 OpenShift 버전마다 변경될 수 있습니다. 검토하고 따르세요"Red Hat의 OpenShift 배포 가이드" 귀하의 환경에 맞게 구성되도록 귀하의 버전에 맞게 구성하세요.

데이터 보호 및 재해 복구

Trident 및 Trident 사용하여 생성된 볼륨에 대한 보호 및 복구 옵션에 대해 알아보세요. 지속성 요구 사항이 있는 각 애플리케이션에 대해 데이터 보호 및 복구 전략이 있어야 합니다.

Trident 복제 및 복구

재해 발생 시 Trident 복구하기 위한 백업을 만들 수 있습니다.

Trident 복제

Trident Kubernetes CRD를 사용하여 자체 상태를 저장하고 관리하고 Kubernetes 클러스터 etcd를 사용하여 메타데이터를 저장합니다.

단계

1. Kubernetes 클러스터 etcd를 사용하여 백업합니다."Kubernetes: etcd 클러스터 백업" .
2. FlexVol volume 에 백업 아티팩트 배치

> NetApp FlexVol 있는 SVM을 다른 SVM과의 SnapMirror 관계로 보호하는 것이 좋습니다.

Trident 회수

Kubernetes CRD와 Kubernetes 클러스터 etcd 스냅샷을 사용하면 Trident 복구할 수 있습니다.

단계

1. 대상 SVM에서 Kubernetes etcd 데이터 파일과 인증서가 포함된 볼륨을 마스터 노드로 설정될 호스트에 마운트합니다.
2. Kubernetes 클러스터에 관련된 모든 필수 인증서를 아래에 복사하세요. `/etc/kubernetes/pki` 그리고 etcd 멤버 파일은 다음과 같습니다. `/var/lib/etcd` .
3. etcd 백업을 사용하여 Kubernetes 클러스터를 복원합니다."Kubernetes: etcd 클러스터 복원" .
4. 달리다 `kubectl get crd` 모든 Trident 사용자 정의 리소스가 나타났는지 확인하고 Trident 객체를 검색하여 모든 데이터를 사용할 수 있는지 확인합니다.

SVM 복제 및 복구

Trident 복제 관계를 구성할 수 없지만 스토리지 관리자는 다음을 사용할 수 있습니다. "ONTAP SnapMirror" SVM을 복제합니다.

재해가 발생하면 SnapMirror 대상 SVM을 활성화하여 데이터 제공을 시작할 수 있습니다. 시스템이 복구되면 기본으로 다시 전환할 수 있습니다.

이 작업에 관하여

SnapMirror SVM 복제 기능을 사용할 때 다음 사항을 고려하세요.

- SVM-DR이 활성화된 각 SVM에 대해 별도의 백엔드를 만들어야 합니다.
- SVM-DR을 지원하는 백엔드에 복제가 필요 없는 볼륨이 프로비저닝되는 것을 방지하기 위해 필요할 때만 복제된 백엔드를 선택하도록 스토리지 클래스를 구성합니다.
- 애플리케이션 관리자는 복제와 관련된 추가 비용과 복잡성을 이해하고 이 프로세스를 시작하기 전에 복구 계획을 신중하게 고려해야 합니다.

SVM 복제

사용할 수 있습니다"ONTAP: SnapMirror SVM 복제" SVM 복제 관계를 생성합니다.

SnapMirror 사용하면 복제할 내용을 제어하는 옵션을 설정할 수 있습니다. 수행할 때 선택한 옵션을 알아야 합니다 .Trident 사용한 SVM 복구 .

- "-동일성-참 유지"SVM 구성 전체를 복제합니다.
- "-discard-configs 네트워크"LIF 및 관련 네트워크 설정은 제외됩니다.
- "-신원-보존 거짓"볼륨과 보안 구성만 복제합니다.

Trident 사용한 SVM 복구

Trident SVM 오류를 자동으로 감지하지 않습니다. 재해가 발생하면 관리자는 수동으로 Trident 장애 조치를 새 SVM으로 시작할 수 있습니다.

단계

1. 예약된 SnapMirror 전송과 진행 중인 SnapMirror 전송을 취소하고, 복제 관계를 끊고, 소스 SVM을 중지한 다음 SnapMirror 대상 SVM을 활성화합니다.
2. 당신이 지정한 경우 `-identity-preserve false` 또는 `-discard-config network` SVM 복제를 구성할 때 다음을 업데이트하세요. `managementLIF` 그리고 `dataLIF` Trident 백엔드 정의 파일에서.
3. 확인하다 `storagePrefix` Trident 백엔드 정의 파일에 있습니다. 이 매개변수는 변경할 수 없습니다. 생략 `storagePrefix` 백엔드 업데이트가 실패하게 됩니다.
4. 다음을 사용하여 새로운 대상 SVM 이름을 반영하도록 모든 필수 백엔드를 업데이트합니다.

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. 당신이 지정한 경우 `-identity-preserve false` 또는 `discard-config network` 모든 애플리케이션 포드를 반송해야 합니다.

> 당신이 지정한 경우 `-identity-preserve true` Trident 에서 프로비저닝한 모든 볼륨은 대상 SVM이 활성화되면 데이터 제공을 시작합니다.

볼륨 복제 및 복구

Trident SnapMirror 복제 관계를 구성할 수 없지만 스토리지 관리자는 다음을 사용할 수 있습니다."ONTAP SnapMirror 복제 및 복구" Trident 에서 생성된 볼륨을 복제합니다.

그런 다음 복구된 볼륨을 Trident 로 가져올 수 있습니다."tridentctl 볼륨 가져오기" .

> 가져오기가 지원되지 않습니다. `ontap-nas-economy` , `ontap-san-economy` , 또는 `ontap-flexgroup-economy` 운전자.

스냅샷 데이터 보호

다음을 사용하여 데이터를 보호하고 복원할 수 있습니다.

- 영구 볼륨(PV)의 Kubernetes 볼륨 스냅샷을 생성하기 위한 외부 스냅샷 컨트롤러와 CRD입니다.

  "볼륨 스냅샷"

- ONTAP 스냅샷을 사용하면 볼륨의 전체 내용을 복원하거나 개별 파일이나 LUN을 복구할 수 있습니다.

  "ONTAP 스냅샷"

보안

보안

여기에 나열된 권장 사항을 사용하여 Trident 설치가 안전한지 확인하세요.

Trident 자체 네임스페이스에서 실행

안정적인 저장소를 보장하고 잠재적인 악성 활동을 차단하려면 애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션이 Trident 객체 정의나 포드에 액세스하지 못하도록 하는 것이 중요합니다.

다른 애플리케이션과 사용자를 Trident 에서 분리하려면 항상 Trident 자체 Kubernetes 네임스페이스에 설치하십시오 .(`trident`). Trident 자체 네임스페이스에 배치하면 Kubernetes 관리 담당자만 Trident 포드와 네임스페이스 CRD 객체에 저장된 아티팩트(해당되는 경우 백엔드 및 CHAP 비밀 등)에 액세스할 수 있습니다. 관리자만 Trident 네임스페이스에 액세스할 수 있도록 허용해야 하며 이를 통해 액세스할 수 있습니다. `tridentctl` 애플리케이션.

ONTAP SAN 백엔드에서 **CHAP** 인증 사용

Trident ONTAP SAN 워크로드에 대한 CHAP 기반 인증을 지원합니다(사용 `ontap-san` 그리고 `ontap-san-economy` 운전자). NetApp 호스트와 스토리지 백엔드 간 인증을 위해 Trident 와 함께 양방향 CHAP를 사용할 것을 권장합니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Trident 양방향 CHAP를 설정하고 CHAP 사용자 이름과 비밀번호를 관리할 수 있습니다. `tridentctl` . 참조하다"ONTAP SAN 드라이버로 백엔드 구성을 준비합니다." Trident ONTAP 백엔드에서 CHAP를 구성하는 방법을 이해합니다.

NetApp HCI 및 **SolidFire** 백엔드에서 **CHAP** 인증 사용

NetApp 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하기 위해 양방향 CHAP를 배포할 것을 권장합니다. Trident 테넌트당 두 개의 CHAP 암호가 포함된 비밀 객체를 사용합니다. Trident 가 설치되면 CHAP 비밀을 관리하고 저장합니다. `tridentvolume` 해당 PV에 대한 CR 객체입니다. PV를 생성하면 Trident CHAP 비밀을 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

Trident 에서 생성된 볼륨은 어떤 볼륨 액세스 그룹과도 연관되지 않습니다.

NVE 및 **NAE**와 함께 **Trident** 사용

NetApp ONTAP 디스크가 도난당하거나 반환되거나 다른 용도로 사용되는 경우 중요한 데이터를 보호하기 위해 저장

데이터 암호화를 제공합니다. 자세한 내용은 다음을 참조하세요."NetApp 볼륨 암호화 구성 개요" .

- 백엔드에서 NAE가 활성화된 경우 Trident 에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다.
  - NVE 암호화 플래그를 설정할 수 있습니다. "" NAE 지원 볼륨을 생성합니다.
- 백엔드에서 NAE가 활성화되지 않은 경우 NVE 암호화 플래그가 설정되지 않는 한 Trident 에서 프로비저닝된 모든 볼륨은 NVE가 활성화됩니다. `false` (기본값) 백엔드 구성에서.

> NAE 지원 백엔드의 Trident 에서 생성된 볼륨은 NVE 또는 NAE로 암호화되어야 합니다.
>
> - NVE 암호화 플래그를 설정할 수 있습니다. `true` Trident 백엔드 구성에서 NAE 암호화를 재정의하고 볼륨별로 특정 암호화 키를 사용합니다.
> - NVE 암호화 플래그를 다음으로 설정 `false` NAE 지원 백엔드에서는 NAE 지원 볼륨을 생성합니다. NVE 암호화 플래그를 설정하여 NAE 암호화를 비활성화할 수 없습니다. `false` .

- NVE 암호화 플래그를 명시적으로 설정하여 Trident 에서 NVE 볼륨을 수동으로 생성할 수 있습니다. `true` .

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하세요.

- "ONTAP SAN 구성 옵션"
- "ONTAP NAS 구성 옵션"

Linux 통합 키 설정(LUKS)

Trident 에서 ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨을 암호화하기 위해 Linux Unified Key Setup(LUKS)을 활성화할 수 있습니다. Trident LUKS로 암호화된 볼륨에 대한 암호 문구 순환과 볼륨 확장을 지원합니다.

Trident 에서 LUKS 암호화 볼륨은 권장하는 대로 aes-xts-plain64 암호 및 모드를 사용합니다."미국 국립표준기술원(NIST)" .

> ASA r2 시스템에서는 LUKS 암호화가 지원되지 않습니다. ASA r2 시스템에 대한 정보는 다음을 참조하세요."ASA r2 스토리지 시스템에 대해 알아보세요" .

시작하기 전에

- 작업자 노드에는 cryptsetup 2.1 이상(3.0 미만)이 설치되어 있어야 합니다. 자세한 내용은 다음을 방문하세요 ."Gitlab: cryptsetup" .
- 성능상의 이유로 NetApp 작업자 노드가 AES-NI(Advanced Encryption Standard New Instructions)를 지원할 것을 권장합니다. AES-NI 지원을 확인하려면 다음 명령을 실행하세요.

```
grep "aes" /proc/cpuinfo
```

아무것도 반환되지 않으면 프로세서가 AES-NI를 지원하지 않는 것입니다. AES-NI에 대한 자세한 내용은 다음을 방문하세요."Intel: 고급 암호화 표준 명령어(AES-NI)" .

LUKS 암호화 활성화

ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨에 대해 Linux Unified Key Setup(LUKS)을 사용하여 볼륨별 호스트 측 암호화를 활성화할 수 있습니다.

단계

1. 백엔드 구성에서 LUKS 암호화 속성을 정의합니다. ONTAP SAN의 백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하세요."ONTAP SAN 구성 옵션" .

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. 사용 `parameters.selector` LUKS 암호화를 사용하여 스토리지 풀을 정의합니다. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. LUKS 암호문구를 포함하는 비밀번호를 생성합니다. 예를 들어:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

제한 사항

LUKS 암호화 볼륨은 ONTAP 중복 제거 및 압축 기능을 활용할 수 없습니다.

LUKS 볼륨 가져오기를 위한 백엔드 구성

LUKS 볼륨을 가져오려면 다음을 설정해야 합니다. `luksEncryption` 에게(`true` 백엔드에서. 그만큼 `luksEncryption` 옵션은 볼륨이 LUKS 규격인지 Trident 알려줍니다.(`true`) 또는 LUKS 규격에 맞지 않음(`false`) 다음 예와 같습니다.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

LUKS 볼륨 가져오기를 위한 **PVC** 구성

LUKS 볼륨을 동적으로 가져오려면 주석을 설정하세요. `trident.netapp.io/luksEncryption` 에게 `true` 이 예에서 보여지는 것처럼 PVC에 LUKS 지원 스토리지 클래스를 포함합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

LUKS 암호 문구 회전

LUKS 암호를 교체하고 교체를 확인할 수 있습니다.

> 볼륨, 스냅샷 또는 비밀에서 더 이상 참조되지 않는다는 것을 확인할 때까지 암호를 잊지 마세요. 참조된 암호문구가 손실되면 볼륨을 마운트할 수 없고 데이터는 암호화되어 액세스할 수 없게 됩니다.

이 작업에 관하여

LUKS 암호 문구 순환은 새로운 LUKS 암호 문구가 지정된 후 볼륨을 마운트하는 포드가 생성될 때 발생합니다. 새로운 포드가 생성되면 Trident 볼륨의 LUKS 암호를 비밀의 활성 암호와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 회전이 발생합니다.
- 볼륨의 암호가 비밀의 활성 암호와 일치하는 경우 `previous-luks-passphrase` 매개변수는 무시됩니다.

단계

1. 추가하세요 `node-publish-secret-name` 그리고 `node-publish-secret-namespace` StorageClass 매개변수. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. 볼륨이나 스냅샷에 있는 기존 암호를 식별합니다.

   용량

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

   스냅샷

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. 볼륨의 LUKS 비밀번호를 업데이트하여 새 암호와 이전 암호구를 지정합니다. 보장하다 `previous-luke-passphrase-name` 그리고 `previous-luks-passphrase` 이전 암호문구와 일치합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. 볼륨을 마운트하여 새로운 포드를 만듭니다. 이는 회전을 시작하는 데 필요합니다.
5. 암호가 회전되었는지 확인하세요.

   용량

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

스냅샷

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

결과

볼륨과 스냅샷에 새로운 암호문구만 반환되면 암호문구가 회전되었습니다.

> 예를 들어 두 개의 암호가 반환되는 경우 `luksPassphraseNames: ["B", "A"]` , 회전이 완료되지 않았습니다. 새로운 포드를 작동시켜 회전을 완료할 수 있습니다.

볼륨 확장 활성화

LUKS로 암호화된 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. 활성화 `CSINodeExpandSecret` 기능 게이트(베타 1.25+). 참조하다 "Kubernetes 1.25: CSI 볼륨의 노드 기반 확장을 위한 비밀 사용" 자세한 내용은.
2. 추가하세요 `node-expand-secret-name` 그리고 `node-expand-secret-namespace` StorageClass 매개변수. 예를 들어:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

결과

온라인 스토리지 확장을 시작하면 kubelet은 드라이버에 적절한 자격 증명을 전달합니다.

Kerberos 비행 중 암호화

Kerberos 전송 중 암호화를 사용하면 관리되는 클러스터와 스토리지 백엔드 간 트래픽에 대한 암호화를 활성화하여 데이터 액세스 보안을 강화할 수 있습니다.

Trident ONTAP 스토리지 백엔드로 사용할 때 Kerberos 암호화를 지원합니다.

- 온프레미스 **ONTAP** - Trident Red Hat OpenShift 및 업스트림 Kubernetes 클러스터에서 온프레미스 ONTAP 볼륨으로의 NFSv3 및 NFSv4 연결을 통해 Kerberos 암호화를 지원합니다.

NFS 암호화를 사용하는 볼륨을 생성, 삭제, 크기 조정, 스냅샷, 복제, 읽기 전용 복제 및 가져오기할 수 있습니다.

온프레미스 **ONTAP** 볼륨을 사용하여 비행 중 **Kerberos** 암호화 구성

관리형 클러스터와 온프레미스 ONTAP 스토리지 백엔드 간의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

> ⓘ 온프레미스 ONTAP 스토리지 백엔드를 사용하는 NFS 트래픽에 대한 Kerberos 암호화는 다음을 통해서만 지원됩니다. `ontap-nas` 저장 드라이버.

시작하기 전에

- 귀하가 다음에 액세스할 수 있는지 확인하십시오. `tridentctl` 공익사업.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있는지 확인하세요.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있는지 확인하세요.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하도록 ONTAP 스토리지 VM을 준비했는지 확인하세요. 참조하다 "dataLIF에서 Kerberos 활성화" 지침을 보려면.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하세요. NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하세요. "NetApp NFSv4 개선 사항 및 모범 사례 가이드" .

ONTAP 내보내기 정책 추가 또는 수정

기존 ONTAP 내보내기 정책에 규칙을 추가하거나 ONTAP 스토리지 VM 루트 볼륨과 업스트림 Kubernetes 클러스터와 공유되는 모든 ONTAP 볼륨에 대한 Kerberos 암호화를 지원하는 새로운 내보내기 정책을 만들어야 합니다. 추가하는 내보내기 정책 규칙이나 만드는 새로운 내보내기 정책은 다음과 같은 액세스 프로토콜과 액세스 권한을 지원해야 합니다.

접근 프로토콜

NFS, NFSv3, NFSv4 액세스 프로토콜을 사용하여 내보내기 정책을 구성합니다.

접근 세부 정보

볼륨에 대한 요구 사항에 따라 세 가지 Kerberos 암호화 버전 중 하나를 구성할 수 있습니다.

- **Kerberos 5** - (인증 및 암호화)
- **Kerberos 5i** - (신원 보호 기능이 있는 인증 및 암호화)
- **Kerberos 5p** - (신원 및 개인 정보 보호를 통한 인증 및 암호화)

적절한 액세스 권한으로 ONTAP 내보내기 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i와 Kerberos 5p 암호화를 혼합하여 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용합니다.

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 접근
유닉스	활성화됨	활성화됨	활성화됨

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 접근
케르베로스 5i	활성화됨	활성화됨	활성화됨
케르베로스 5p	활성화됨	활성화됨	활성화됨

ONTAP 내보내기 정책과 내보내기 정책 규칙을 만드는 방법에 대한 자세한 내용은 다음 문서를 참조하세요.

- "수출 정책 만들기"
- "내보내기 정책에 규칙 추가"

스토리지 백엔드 생성

Kerberos 암호화 기능을 포함하는 Trident 스토리지 백엔드 구성을 만들 수 있습니다.

이 작업에 관하여

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 다음을 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다. `spec.nfsMountOptions` 매개변수:

- `spec.nfsMountOptions: sec=krb5`(인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5i`(신원 보호 기능이 있는 인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5p`(신원 및 개인 정보 보호를 통한 인증 및 암호화)

Kerberos 수준을 하나만 지정하세요. 매개변수 목록에서 두 개 이상의 Kerberos 암호화 수준을 지정하는 경우 첫 번째 옵션만 사용됩니다.

단계

1. 관리되는 클러스터에서 다음 예를 사용하여 스토리지 백엔드 구성 파일을 만듭니다. <> 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 이전 단계에서 만든 구성 파일을 사용하여 백엔드를 만듭니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 생성

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하기 위해 스토리지 클래스를 생성할 수 있습니다.

이 작업에 관하여

저장소 클래스 객체를 생성할 때 다음을 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다. `mountOptions` 매개변수:

- `mountOptions: sec=krb5`(인증 및 암호화)
- `mountOptions: sec=krb5i`(신원 보호 기능이 있는 인증 및 암호화)
- `mountOptions: sec=krb5p`(신원 및 개인 정보 보호를 통한 인증 및 암호화)

Kerberos 수준을 하나만 지정하세요. 매개변수 목록에서 두 개 이상의 Kerberos 암호화 수준을 지정하는 경우 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 개체에서 지정한 수준과 다른 경우 스토리지 클래스 개체가 우선합니다.

단계

1. 다음 예를 사용하여 StorageClass Kubernetes 객체를 만듭니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. 저장 클래스를 만듭니다.

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 스토리지 클래스가 생성되었는지 확인하세요.

```
kubectl get sc ontap-nas-sc
```

다음과 비슷한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

공급량

스토리지 백엔드와 스토리지 클래스를 만든 후 이제 볼륨을 프로비저닝할 수 있습니다. 지침은 다음을 참조하세요. "볼륨 제공" .

Azure NetApp Files 볼륨을 사용하여 진행 중인 Kerberos 암호화 구성

관리되는 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 간의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리되는 Red Hat OpenShift 클러스터에서 Trident 활성화했는지 확인하세요.
- 귀하가 다음에 액세스할 수 있는지 확인하십시오. `tridentctl` 공익사업.
- 요구 사항을 확인하고 다음 지침을 따르면 Kerberos 암호화를 위한 Azure NetApp Files 저장소 백엔드가 준비되었는지 확인할 수 있습니다. "Azure NetApp Files 설명서" .
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하세요. NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하세요. "NetApp NFSv4 개선 사항 및 모범 사례 가이드" .

스토리지 백엔드 생성

Kerberos 암호화 기능을 포함하는 Azure NetApp Files 스토리지 백엔드 구성을 만들 수 있습니다.

이 작업에 관하여

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 다음 두 가지 수준 중 하나에 적용되도록 정의할 수 있습니다.

- *저장소 백엔드 수준*을 사용합니다. `spec.kerberos` 필드
- *가상 풀 레벨*을 사용하여 `spec.storage.kerberos` 필드

가상 풀 수준에서 구성을 정의하는 경우 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

어느 수준에서든 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다.

- `kerberos: sec=krb5`(인증 및 암호화)
- `kerberos: sec=krb5i`(신원 보호 기능이 있는 인증 및 암호화)
- `kerberos: sec=krb5p`(신원 및 개인 정보 보호를 통한 인증 및 암호화)

단계

1. 관리되는 클러스터에서 스토리지 백엔드를 정의해야 하는 위치(스토리지 백엔드 수준 또는 가상 풀 수준)에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 만듭니다. <> 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

스토리지 백엔드 수준 예제

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

가상 풀 레벨 예시

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. 이전 단계에서 만든 구성 파일을 사용하여 백엔드를 만듭니다.

```
tridentctl create backend -f <backend-configuration-file>
```

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하면 로그를 보고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 파악하고 수정한 후에는 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스 생성

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하기 위해 스토리지 클래스를 생성할 수 있습니다.

단계

1. 다음 예를 사용하여 StorageClass Kubernetes 객체를 만듭니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. 저장 클래스를 만듭니다.

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. 스토리지 클래스가 생성되었는지 확인하세요.

```
kubectl get sc -sc-nfs
```

다음과 비슷한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

공급량

스토리지 백엔드와 스토리지 클래스를 만든 후 이제 볼륨을 프로비저닝할 수 있습니다. 지침은 다음을 참조하세요. "볼륨 제공" .

Trident Protect로 애플리케이션을 보호하세요

Trident Protect에 대해 알아보세요

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너가 지원하는 상태 저장 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다. Trident Protect는 퍼블릭 클라우드와 온프레미스 환경에서 컨테이너화된 워크로드의 관리, 보호 및 이동을 간소화합니다. 또한 API와 CLI를 통해 자동화 기능도 제공합니다.

Trident Protect를 사용하면 사용자 정의 리소스(CR)를 생성하거나 Trident Protect CLI를 사용하여 애플리케이션을 보호할 수 있습니다.

다음은 무엇인가요?

Trident Protect를 설치하기 전에 요구 사항에 대해 알아볼 수 있습니다.

- "Trident 프로텍트 요구 사항"

Trident Protect 설치

Trident 프로텍트 요구 사항

운영 환경, 애플리케이션 클러스터, 애플리케이션 및 라이선스의 준비 상태를 확인하여 시작하세요. Trident Protect를 배포하고 운영하려면 환경이 이러한 요구 사항을 충족하는지 확인하세요.

Trident Protect Kubernetes 클러스터 호환성

Trident Protect는 다음을 포함한 광범위한 완전 관리형 및 자체 관리형 Kubernetes 제품과 호환됩니다.

- 아마존 엘라스틱 쿠버네티스 서비스(EKS)
- Google Kubernetes 엔진(GKE)
- Microsoft Azure 쿠버네티스 서비스(AKS)
- 레드햇 오픈시프트
- SUSE Rancher
- VMware Tanzu 포트폴리오
- 업스트림 쿠버네티스

- Trident Protect 백업은 Linux 컴퓨트 노드에서만 지원됩니다. Windows 컴퓨팅 노드에서는 백업 작업이 지원되지 않습니다.
- Trident Protect를 설치하는 클러스터가 실행 중인 스냅샷 컨트롤러와 관련 CRD로 구성되어 있는지 확인하세요. 스냅샷 컨트롤러를 설치하려면 다음을 참조하세요. "이 지침" .

Trident Protect 스토리지 백엔드 호환성

Trident Protect는 다음과 같은 스토리지 백엔드를 지원합니다.

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP 스토리지 어레이
- Google Cloud NetApp Volumes
- Azure NetApp Files

스토리지 백엔드가 다음 요구 사항을 충족하는지 확인하세요.

- 클러스터에 연결된 NetApp 스토리지가 Trident 24.02 이상(Trident 24.10 권장)을 사용하는지 확인하세요.
- NetApp ONTAP 스토리지 백엔드가 있는지 확인하세요.
- 백업을 저장하기 위해 개체 스토리지 버킷을 구성했는지 확인하세요.
- 애플리케이션이나 애플리케이션 데이터 관리 작업에 사용할 애플리케이션 네임스페이스를 만듭니다. Trident Protect는 이러한 네임스페이스를 자동으로 생성하지 않습니다. 사용자 정의 리소스에 존재하지 않는 네임스페이스를 지정하면 작업이 실패합니다.

nas-economy 볼륨에 대한 요구 사항

Trident Protect는 NAS 경제형 볼륨에 대한 백업 및 복원 작업을 지원합니다. 현재 NAS 경제 볼륨에 대한 스냅샷, 복제 및 SnapMirror 복제는 지원되지 않습니다. Trident Protect와 함께 사용하려는 각 NAS-Economy 볼륨에 대해 스냅샷 디렉토리를 활성화해야 합니다.

일부 애플리케이션은 스냅샷 디렉토리를 사용하는 볼륨과 호환되지 않습니다. 이러한 애플리케이션의 경우 ONTAP 스토리지 시스템에서 다음 명령을 실행하여 스냅샷 디렉토리를 숨겨야 합니다.

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

다음 명령을 각 nas-economy 볼륨에 대해 실행하여 스냅샷 디렉토리를 활성화할 수 있습니다. `<volume-UUID>` 변경하려는 볼륨의 UUID를 사용하여:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Trident 백엔드 구성 옵션을 설정하여 새 볼륨에 대해 기본적으로 스냅샷 디렉토리를 활성화할 수 있습니다. `snapshotDir` 에게 `true` . 기존 볼륨에는 영향을 미치지 않습니다.

KubeVirt VM을 사용하여 데이터 보호

Trident Protect 24.10 및 24.10.1 이상 버전은 KubeVirt VM에서 실행되는 애플리케이션을 보호할 때 동작이 다릅니다. 두 버전 모두 데이터 보호 작업 중에 파일 시스템 동결 및 동결 해제를 활성화하거나 비활성화할 수 있습니다.

복원 작업 중에 `VirtualMachineSnapshots` 가상 머신(VM)에 대해 생성된 항목은 복원되지 않습니다.

Trident 프로텍트 **24.10**

Trident Protect 24.10은 데이터 보호 작업 중에 KubeVirt VM 파일 시스템의 일관된 상태를 자동으로 보장하지 않습니다. Trident Protect 24.10을 사용하여 KubeVirt VM 데이터를 보호하려면 데이터 보호 작업을 시작하기 전에 파일 시스템의 동결/동결 해제 기능을 수동으로 활성화해야 합니다. 이렇게 하면 파일 시스템이 일관된 상태를 유지하게 됩니다.

데이터 보호 작업 중 VM 파일 시스템의 동결 및 동결 해제를 관리하도록 Trident Protect 24.10을 구성할 수 있습니다. "가상화 구성" 그리고 다음 명령을 사용합니다.

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Trident Protect 24.10.1 이상

Trident Protect 24.10.1부터 Trident Protect는 데이터 보호 작업 중에 KubeVirt 파일 시스템을 자동으로 동결 및 동결 해제합니다. 선택적으로 다음 명령을 사용하여 이 자동 동작을 비활성화할 수 있습니다.

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

SnapMirror 복제 요구 사항

NetApp SnapMirror 복제는 다음 ONTAP 솔루션에 대해 Trident Protect와 함께 사용할 수 있습니다.

- 온프레미스 NetApp FAS, AFF 및 ASA 클러스터
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

SnapMirror 복제를 위한 **ONTAP** 클러스터 요구 사항

SnapMirror 복제를 사용하려면 ONTAP 클러스터가 다음 요구 사항을 충족하는지 확인하세요.

- * NetApp Trident*: NetApp Trident 백엔드로 ONTAP 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 존재해야 합니다. Trident Protect는 다음 드라이버로 지원되는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 복제를 지원합니다.
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)
- 라이선스: 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이선스는 소스 및 대상 ONTAP 클러스터 모두에서 활성화되어야 합니다. 참조하다 "ONTAP 의 SnapMirror 라이선싱 개요" 자세한 내용은.

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

SnapMirror 비동기 보호만 지원됩니다.

SnapMirror 복제를 위한 피어링 고려 사항

스토리지 백엔드 피어링을 사용하려면 환경이 다음 요구 사항을 충족하는지 확인하세요.

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

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

- * NetApp Trident 및 SVM*: 피어링된 원격 SVM은 대상 클러스터의 NetApp Trident 에서 사용할 수 있어야 합니다.
- 관리형 백엔드: 복제 관계를 생성하려면 Trident Protect에서 ONTAP 스토리지 백엔드를 추가하고 관리해야 합니다.

SnapMirror 복제를 위한 Trident / ONTAP 구성

Trident Protect를 사용하려면 소스 및 대상 클러스터 모두에 대한 복제를 지원하는 하나 이상의 스토리지 백엔드를 구성해야 합니다. 소스 및 대상 클러스터가 동일한 경우, 최상의 복원력을 위해 대상 애플리케이션은 소스 애플리케이션과 다른 스토리지 백엔드를 사용해야 합니다.

SnapMirror 복제를 위한 Kubernetes 클러스터 요구 사항

Kubernetes 클러스터가 다음 요구 사항을 충족하는지 확인하세요.

- **AppVault** 접근성: 애플리케이션 개체 복제를 위해 소스 클러스터와 대상 클러스터 모두 AppVault에서 읽고 쓸 수 있는 네트워크 액세스 권한이 있어야 합니다.
- 네트워크 연결: WAN을 통해 클러스터와 AppVault 간의 통신을 활성화하기 위해 방화벽 규칙, 버킷 권한 및 IP 허용 목록을 구성합니다.

많은 기업 환경에서는 WAN 연결 전반에 걸쳐 엄격한 방화벽 정책을 구현합니다. 복제를 구성하기 전에 인프라 팀과 함께 이러한 네트워크 요구 사항을 확인하세요.

Trident Protect 설치 및 구성

사용자 환경이 Trident Protect에 대한 요구 사항을 충족하는 경우 다음 단계에 따라 클러스터에 Trident Protect를 설치할 수 있습니다. NetApp 에서 Trident Protect를 구입하거나 개인 레지스트리에서 설치할 수 있습니다. 클러스터가 인터넷에 액세스할 수 없는 경우 개인 레지스트리에서 설치하는 것이 도움이 됩니다.

Trident Protect 설치

NetApp 에서 **Trident Protect** 설치

단계

1. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

2. Helm을 사용하여 Trident Protect를 설치하세요. 바꾸다 `<name-of-cluster>` 클러스터에 할당되고 클러스터의 백업과 스냅샷을 식별하는 데 사용되는 클러스터 이름이 포함됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2506.0 --create
-namespace --namespace trident-protect
```

개인 레지스트리에서 **Trident Protect** 설치

Kubernetes 클러스터가 인터넷에 액세스할 수 없는 경우 개인 이미지 레지스트리에서 Trident Protect를 설치할 수 있습니다. 다음 예에서 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

단계

1. 다음 이미지를 로컬 머신으로 가져와서 태그를 업데이트한 다음 개인 레지스트리에 푸시합니다.

```
netapp/controller:25.06.0
netapp/restic:25.06.0
netapp/kopia:25.06.0
netapp/trident-autosupport:25.06.0
netapp/exechook:25.06.0
netapp/resourcebackup:25.06.0
netapp/resourcerestore:25.06.0
netapp/resourcedelete:25.06.0
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

   예를 들어:

```
docker pull netapp/controller:25.06.0
```

```
docker tag netapp/controller:25.06.0 <private-registry-
url>/controller:25.06.0
```

```
docker push <private-registry-url>/controller:25.06.0
```

2. Trident Protect 시스템 네임스페이스를 만듭니다.

```
kubectl create ns trident-protect
```

3. 레지스트리에 로그인하세요:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 개인 레지스트리 인증에 사용할 풀 시크릿을 만듭니다.

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. Trident Helm 저장소를 추가합니다.

```
helm repo add netapp-trident-protect https://netapp.github.io/trident-protect-helm-chart
```

6. 라는 이름의 파일을 만듭니다. `protectValues.yaml` . 다음 Trident Protect 설정이 포함되어 있는지 확인하세요.

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. Helm을 사용하여 Trident Protect를 설치하세요. 바꾸다 `<name_of_cluster>` 클러스터에 할당되고 클러스터의 백업과 스냅샷을 식별하는 데 사용되는 클러스터 이름이 포함됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect --set clusterName=<name_of_cluster> --version 100.2506.0 --create-namespace --namespace trident-protect -f protectValues.yaml
```

Trident Protect CLI 플러그인을 설치하세요

Trident Protect 명령줄 플러그인을 사용할 수 있습니다. 이 플러그인은 Trident 의 확장 기능입니다. `tridentctl` Trident Protect 사용자 정의 리소스(CR)를 생성하고 상호 작용할 수 있는 유틸리티입니다.

Trident Protect CLI 플러그인을 설치하세요

명령줄 유틸리티를 사용하려면 먼저 클러스터에 액세스하는 데 사용하는 컴퓨터에 유틸리티를 설치해야 합니다. 컴퓨터에서 x64 또는 ARM CPU를 사용하는지 여부에 따라 다음 단계를 따르세요.

Linux AMD64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-linux-amd64
```

Linux ARM64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-linux-arm64
```

Mac AMD64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-macos-amd64
```

Mac ARM64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.06.0/tridentctl-protect-macos-arm64
```

1. 플러그인 바이너리에 대한 실행 권한을 활성화합니다.

```
chmod +x tridentctl-protect
```

2. 플러그인 바이너리를 PATH 변수에 정의된 위치에 복사합니다. 예를 들어, `/usr/bin` 또는 `/usr/local/bin` (높은 권한이 필요할 수 있음):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 선택적으로 플러그인 바이너리를 홈 디렉토리의 원하는 위치에 복사할 수 있습니다. 이 경우 해당 위치가 PATH 변수의 일부인지 확인하는 것이 좋습니다.

```
cp ./tridentctl-protect ~/bin/
```

플러그인을 PATH 변수의 위치에 복사하면 다음을 입력하여 플러그인을 사용할 수 있습니다. `tridentctl-protect` 또는 `tridentctl protect` 어느 위치에서나.

Trident CLI 플러그인 도움말 보기

플러그인의 기능에 대한 자세한 도움말을 얻으려면 내장된 플러그인 도움말 기능을 사용할 수 있습니다.

단계

1. 도움말 기능을 사용하여 사용 지침을 확인하세요.

```
tridentctl-protect help
```

명령 자동 완성 활성화

Trident Protect CLI 플러그인을 설치한 후 특정 명령에 대한 자동 완성을 활성화할 수 있습니다.

Bash 셸에 대한 자동 완성을 활성화합니다.

단계

1. 완성 스크립트를 다운로드하세요:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.bash
```

2. 스크립트를 포함할 새 디렉토리를 홈 디렉토리에 만듭니다.

```
mkdir -p ~/.bash/completions
```

3. 다운로드한 스크립트를 다음으로 이동합니다. `~/.bash/completions` 예배 규칙서:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 다음 줄을 추가하세요 `~/.bashrc` 홈 디렉토리에 있는 파일:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Z 셸에 대한 자동 완성을 활성화합니다.

단계

1. 완성 스크립트를 다운로드하세요:

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.zsh
```

2. 스크립트를 포함할 새 디렉토리를 홈 디렉토리에 만듭니다.

```
mkdir -p ~/.zsh/completions
```

3. 다운로드한 스크립트를 다음으로 이동합니다. `~/.zsh/completions` 예배 규칙서:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 다음 줄을 추가하세요 `~/.zprofile` 홈 디렉토리에 있는 파일:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

결과

다음 셸 로그인 시 tridentctl-protect 플러그인을 사용하여 명령 자동 완성 기능을 사용할 수 있습니다.

Trident Protect 설치 사용자 정의

Trident Protect의 기본 구성을 사용자 환경의 특정 요구 사항에 맞게 사용자 정의할 수 있습니다.

Trident Protect 컨테이너 리소스 제한 지정

Trident Protect를 설치한 후에는 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 리소스 제한을 지정할 수 있습니다. 리소스 제한을 설정하면 Trident Protect 작업에서 클러스터 리소스가 얼마나 소모되는지 제어할 수 있습니다.

단계

1. 라는 이름의 파일을 만듭니다. `resourceLimits.yaml` .
2. 사용자 환경의 요구 사항에 따라 Trident Protect 컨테이너에 대한 리소스 제한 옵션으로 파일을 채웁니다.

   다음 예제 구성 파일은 사용 가능한 설정을 보여주고 각 리소스 제한에 대한 기본값을 포함합니다.

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
```

```
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. 값을 적용합니다 `resourceLimits.yaml` 파일:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

보안 컨텍스트 제약 조건 사용자 정의

Trident Protect를 설치한 후에는 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 OpenShift 보안 컨텍스트 제약 조건(SCC)을 수정할 수 있습니다. 이러한 제약 조건은 Red Hat OpenShift 클러스터의 포드에 대한 보안 제한 사항을 정의합니다.

단계

1. 라는 이름의 파일을 만듭니다. `sccconfig.yaml` .
2. 파일에 SCC 옵션을 추가하고 환경의 요구 사항에 맞게 매개변수를 수정합니다.

   다음 예에서는 SCC 옵션에 대한 매개변수의 기본값을 보여줍니다.

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

이 표에서는 SCC 옵션의 매개변수를 설명합니다.

매개변수	설명	기본
만들다	SCC 리소스를 생성할 수 있는지 여부를 결정합니다. SCC 리소스는 다음 경우에만 생성됩니다. `scc.create` 로 설정됩니다 `true` Helm 설치 프로세스는 OpenShift 환경을 식별합니다. OpenShift에서 작동하지 않는 경우 또는 `scc.create` 로 설정됩니다 `false` , SCC 리소스가 생성되지 않습니다.	true
이름	SCC의 이름을 지정합니다.	삼지창 보호 직업
우선 사항	SCC의 우선순위를 정의합니다. 우선순위 값이 높은 SCC는 값이 낮은 SCC보다 먼저 평가됩니다.	1

3. 값을 적용합니다 `sccconfig.yaml` 파일:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect -f
sccconfig.yaml --reuse-values
```

이렇게 하면 기본값이 다음에 지정된 값으로 대체됩니다. `sccconfig.yaml` 파일.

추가 **Trident Protect** 헬름 차트 설정 구성

사용자의 특정 요구 사항에 맞게 AutoSupport 설정과 네임스페이스 필터링을 사용자 정의할 수 있습니다. 다음 표에서는 사용 가능한 구성 매개변수를 설명합니다.

매개변수	유형	설명
autoSupport.proxy	끈	NetApp AutoSupport 연결을 위한 프록시 URL을 구성합니다. 이를 사용하면 프록시 서버를 통해 지원 번들 업로드를 라우팅할 수 있습니다. 예: `http://my.proxy.url` .
autoSupport.안전하지 않음	부울	AutoSupport 프록시 연결에 대한 TLS 검증을 건너뜁니다. `true` . 안전하지 않은 프록시 연결에만 사용하세요. (기본: `false`)

매개변수	유형	설명
autoSupport.활성화됨	부울	일일 Trident Protect AutoSupport 번들 업로드를 활성화하거나 비활성화합니다. 설정 시 `false` , 예약된 일일 업로드는 비활성화되지만, 여전히 수동으로 지원 번들을 생성할 수 있습니다. (기본: `true`)
restoreSkipNamespaceAnnotations	끈	백업 및 복원 작업에서 제외할 네임스페이스 주석의 쉼표로 구분된 목록입니다. 주석을 기준으로 네임스페이스를 필터링할 수 있습니다.
restoreSkipNamespaceLabels	끈	백업 및 복원 작업에서 제외할 네임스페이스 레이블의 쉼표로 구분된 목록입니다. 라벨을 기준으로 네임스페이스를 필터링할 수 있습니다.

YAML 구성 파일이나 명령줄 플래그를 사용하여 이러한 옵션을 구성할 수 있습니다.

YAML 파일 사용

단계

1. 구성 파일을 만들고 이름을 지정하세요. `values.yaml` .
2. 생성한 파일에 사용자 정의하려는 구성 옵션을 추가합니다.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. 다음을 채운 후 `values.yaml` 올바른 값을 가진 파일을 만들려면 구성 파일을 적용하세요:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

CLI 플래그 사용

단계

1. 다음 명령을 사용하십시오. `--set` 개별 매개변수를 지정하는 플래그:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set restoreSkipNamespaceAnnotations="annotation1,annotation2" \
  --set restoreSkipNamespaceLabels="label1,label2" \
  --reuse-values
```

Trident Protect 포드를 특정 노드로 제한

Kubernetes nodeSelector 노드 선택 제약 조건을 사용하면 노드 레이블을 기준으로 Trident Protect 포드를 실행할 수 있는 노드를 제어할 수 있습니다. 기본적으로 Trident Protect는 Linux를 실행하는 노드로 제한됩니다. 사용자의 요구 사항에 따라 이러한 제약 조건을 추가로 사용자 정의할 수 있습니다.

단계

1. 라는 이름의 파일을 만듭니다. `nodeSelectorConfig.yaml` .
2. 파일에 nodeSelector 옵션을 추가하고 파일을 수정하여 환경의 요구 사항에 맞게 노드 레이블을 추가하거나 변경하여 제한합니다. 예를 들어, 다음 파일에는 기본 OS 제한이 포함되어 있지만 특정 지역과 앱 이름도 대상으로 합니다.

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. 값을 적용합니다 `nodeSelectorConfig.yaml` 파일:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

이렇게 하면 기본 제한 사항이 사용자가 지정한 제한 사항으로 대체됩니다. `nodeSelectorConfig.yaml` 파일.

Trident Protect 관리

Trident Protect 권한 및 액세스 제어 관리

Trident Protect는 역할 기반 액세스 제어(RBAC)의 Kubernetes 모델을 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스와 연관된 기본 서비스 계정을 제공합니다. 사용자 수가 많거나 특정 보안 요구 사항이 있는 조직인 경우 Trident Protect의 RBAC 기능을 사용하여 리소스와 네임스페이스에 대한 액세스를 보다 세부적으로 제어할 수 있습니다.

클러스터 관리자는 항상 기본 리소스에 액세스할 수 있습니다. `trident-protect` 네임스페이스에 액세스할 수 있으며 다른 모든 네임스페이스의 리소스에도 액세스할 수 있습니다. 리소스와 애플리케이션에 대한 액세스를 제어하려면 추가 네임스페이스를 만들고 해당 네임스페이스에 리소스와 애플리케이션을 추가해야 합니다.

기본적으로 사용자는 애플리케이션 데이터 관리 CR을 생성할 수 없습니다. `trident-protect` 네임스페이스. 애플리케이션 네임스페이스에 애플리케이션 데이터 관리 CR을 만들어야 합니다(가장 좋은 방법은 연관된 애플리케이션과 동일한 네임스페이스에 애플리케이션 데이터 관리 CR을 만드는 것입니다).

> 다음을 포함하는 권한이 있는 Trident Protect 사용자 정의 리소스 개체에 대한 액세스 권한은 관리자에게만 부여됩니다.
>
> - **AppVault**: 버킷 자격 증명 데이터가 필요합니다.
> - **AutoSupportBundle**: 메트릭, 로그 및 기타 중요한 Trident Protect 데이터를 수집합니다.
> - **AutoSupportBundleSchedule**: 로그 수집 일정을 관리합니다.
>
> 가장 좋은 방법은 RBAC를 사용하여 권한이 있는 개체에 대한 액세스를 관리자로 제한하는 것입니다.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 다음을 참조하세요. "Kubernetes RBAC 문서" .

서비스 계정에 대한 자세한 내용은 다음을 참조하세요. "Kubernetes 서비스 계정 문서" .

예: 두 그룹의 사용자에 대한 액세스 관리

예를 들어, 어떤 조직에 클러스터 관리자, 엔지니어링 사용자 그룹, 마케팅 사용자 그룹이 있다고 가정해 보겠습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각자의 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 만들기 위해 다음 작업을 완료합니다.

1단계**:** 각 그룹의 리소스를 포함할 네임스페이스 만들기

네임스페이스를 만들면 리소스를 논리적으로 분리하고 해당 리소스에 누가 액세스할 수 있는지 더 효과적으로 제어할 수 있습니다.

단계

1. 엔지니어링 그룹에 대한 네임스페이스를 만듭니다.

```
kubectl create ns engineering-ns
```

2. 마케팅 그룹을 위한 네임스페이스를 만듭니다.

```
kubectl create ns marketing-ns
```

2단계**:** 각 네임스페이스의 리소스와 상호 작용하기 위한 새 서비스 계정 만들기

새로 만든 네임스페이스마다 기본 서비스 계정이 제공되지만, 나중에 필요한 경우 그룹 간에 권한을 더욱 세분화할 수 있도록 각 사용자 그룹에 대한 서비스 계정을 만들어야 합니다.

단계

1. 엔지니어링 그룹에 대한 서비스 계정을 만듭니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 마케팅 그룹을 위한 서비스 계정을 만듭니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

3단계**:** 각 새 서비스 계정에 대한 비밀 만들기

서비스 계정 비밀번호는 서비스 계정을 인증하는 데 사용되며, 손상된 경우 쉽게 삭제하고 다시 만들 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 대한 비밀을 만듭니다.

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. 마케팅 서비스 계정에 대한 비밀을 만드세요.

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

4단계**: ClusterRole** 객체를 각각의 새 서비스 계정에 바인딩하기 위해 **RoleBinding** 객체를 만듭니다**.**

Trident Protect를 설치하면 기본 ClusterRole 개체가 생성됩니다. RoleBinding 객체를 생성하고 적용하여 이 ClusterRole을 서비스 계정에 바인딩할 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 ClusterRole을 바인딩합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. ClusterRole을 마케팅 서비스 계정에 바인딩합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

5단계: 권한 테스트

권한이 올바른지 테스트합니다.

단계

1. 엔지니어링 사용자가 엔지니어링 리소스에 액세스할 수 있는지 확인하세요.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. 엔지니어링 사용자가 마케팅 리소스에 액세스할 수 없는지 확인하세요.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

6단계: AppVault 개체에 대한 액세스 권한 부여

백업 및 스냅샷과 같은 데이터 관리 작업을 수행하려면 클러스터 관리자가 개별 사용자에게 AppVault 개체에 대한 액세스 권한을 부여해야 합니다.

단계

1. AppVault 및 비밀번호 조합 YAML 파일을 만들고 적용하여 사용자에게 AppVault에 대한 액세스 권한을 부여합니다. 예를 들어, 다음 CR은 사용자에게 AppVault에 대한 액세스 권한을 부여합니다. `eng-user` :

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. 클러스터 관리자가 네임스페이스의 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 만들고 적용합니다. 예를 들어:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. RoleBinding CR을 생성하고 적용하여 사용자 eng-user에게 권한을 바인딩합니다. 예를 들어:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. 권한이 올바른지 확인하세요.

   a. 모든 네임스페이스에 대한 AppVault 개체 정보를 검색해 보세요.

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   다음과 비슷한 출력이 표시됩니다.

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. 사용자가 이제 액세스 권한이 있는 AppVault 정보를 얻을 수 있는지 테스트합니다.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

다음과 비슷한 출력이 표시됩니다.

```
yes
```

결과

AppVault 권한을 부여한 사용자는 애플리케이션 데이터 관리 작업에 대해 권한이 있는 AppVault 개체를 사용할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 만들 수 없습니다.

Trident Protect 리소스 모니터링

kube-state-metrics, Prometheus, Alertmanager 오픈 소스 도구를 사용하여 Trident Protect로 보호되는 리소스의 상태를 모니터링할 수 있습니다.

kube-state-metrics 서비스는 Kubernetes API 통신에서 메트릭을 생성합니다. Trident Protect와 함께 사용하면 사용자 환경의 리소스 상태에 대한 유용한 정보가 제공됩니다.

Prometheus는 kube-state-metrics에서 생성된 데이터를 수집하고 이러한 객체에 대한 쉽게 읽을 수 있는 정보로 제공할 수 있는 툴킷입니다. kube-state-metrics와 Prometheus를 함께 사용하면 Trident Protect로 관리하는 리소스의 상태와 상태를 모니터링할 수 있는 방법을 제공합니다.

Alertmanager는 Prometheus와 같은 도구에서 보낸 알림을 수집하여 사용자가 구성한 대상으로 라우팅하는 서비스입니다.

> 이 단계에 포함된 구성과 지침은 단지 예시일 뿐입니다. 사용자 환경에 맞게 사용자 정의해야 합니다. 구체적인 지침과 지원에 대해서는 다음 공식 문서를 참조하세요.
>
> - "kube-state-metrics 문서"
> - "프로메테우스 문서"
> - "Alertmanager 문서"

1단계**:** 모니터링 도구 설치

Trident Protect에서 리소스 모니터링을 활성화하려면 kube-state-metrics, Promethus, Alertmanager를 설치하고 구성해야 합니다.

kube-state-metrics 설치

Helm을 사용하여 kube-state-metrics를 설치할 수 있습니다.

단계

1. kube-state-metrics Helm 차트를 추가합니다. 예를 들어:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. 클러스터에 Prometheus ServiceMonitor CRD를 적용합니다.

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Helm 차트에 대한 구성 파일을 만듭니다(예: `metrics-config.yaml`). 다음 예제 구성을 사용자 환경에 맞게 사용자 정의할 수 있습니다.

metrics-config.yaml: kube-state-metrics Helm 차트 구성

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Helm 차트를 배포하여 kube-state-metrics를 설치합니다. 예를 들어:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Trident Protect에서 사용하는 사용자 정의 리소스에 대한 메트릭을 생성하도록 kube-state-metrics를 구성하려면 다음 지침을 따르세요. "kube-state-metrics 사용자 정의 리소스 설명서" .

프로메테우스 설치

다음 지침에 따라 Prometheus를 설치할 수 있습니다. "프로메테우스 문서" .

Alertmanager 설치

다음 지침에 따라 Alertmanager를 설치할 수 있습니다. "Alertmanager 문서" .

2단계: 모니터링 도구를 함께 작동하도록 구성

모니터링 도구를 설치한 후에는 도구가 함께 작동하도록 구성해야 합니다.

단계

1. kube-state-metrics를 Prometheus와 통합합니다. Prometheus 구성 파일 편집(`prometheus.yaml`) 및 kube-state-metrics 서비스 정보를 추가합니다. 예를 들어:

prometheus.yaml: Prometheus와 kube-state-metrics 서비스 통합

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Prometheus를 구성하여 알림을 Alertmanager로 라우팅합니다. Prometheus 구성 파일 편집(`prometheus.yaml`) 다음 섹션을 추가합니다.

prometheus.yaml: Alertmanager에 알림 보내기

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

결과

이제 Prometheus는 kube-state-metrics에서 메트릭을 수집하고 Alertmanager에 알림을 보낼 수 있습니다. 이제 알림을 트리거하는 조건과 알림을 보낼 위치를 구성할 준비가 되었습니다.

3단계**:** 알림 및 알림 대상 구성

도구가 함께 작동하도록 구성한 후에는 어떤 유형의 정보가 알림을 트리거하는지, 알림을 어디로 보낼지 구성해야 합니다.

경고 예**:** 백업 실패

다음 예제에서는 백업 사용자 정의 리소스의 상태가 다음과 같이 설정될 때 트리거되는 중요 경고를 정의합니다. `Error` 5초 이상. 이 예제를 사용자 환경에 맞게 사용자 정의하고 이 YAML 스니펫을 포함할 수 있습니다. `prometheus.yaml` 구성 파일:

rules.yaml: 실패한 백업에 대한 **Prometheus** 알림을 정의합니다**.**

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

다른 채널에 알림을 보내도록 **Alertmanager**를 구성합니다**.**

이메일, PagerDuty, Microsoft Teams 또는 기타 알림 서비스와 같은 다른 채널에 알림을 보내도록 Alertmanager를 구성하려면 해당 구성을 지정해야 합니다. `alertmanager.yaml` 파일.

다음 예제에서는 Alertmanager가 Slack 채널에 알림을 보내도록 구성합니다. 이 예제를 사용자 환경에 맞게 사용자 지정하려면 다음 값을 바꾸십시오. `api_url` 사용자 환경에서 사용되는 Slack 웹훅 URL이 포함된 키:

alertmanager.yaml: Slack 채널에 알림 보내기

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Trident Protect 지원 번들 생성

Trident Protect를 사용하면 관리자는 관리 중인 클러스터와 앱에 대한 로그, 메트릭, 토폴로지 정보 등 NetApp 지원에 유용한 정보가 포함된 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 사용자 정의 리소스(CR) 파일을 사용하여 NetApp 지원 사이트(NSS)에 지원 번들을 업로드할 수 있습니다.

CR을 사용하여 지원 번들을 만듭니다.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml`).
2. 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.triggerType**: (필수) 지원 번들이 즉시 생성되는지, 아니면 예약되는지를 결정합니다. 예약된 번들 생성은 UTC 기준 오전 12시에 이루어집니다. 가능한 값:
     - 예정됨
     - 수동
   - **spec.uploadEnabled**: (선택 사항) 지원 번들이 생성된 후 NetApp 지원 사이트에 업로드할지 여부를 제어합니다. 지정하지 않으면 기본값으로 설정됩니다. `false` . 가능한 값:
     - true
     - false (기본값)
   - **spec.dataWindowStart**: (선택 사항) 지원 번들에 포함된 데이터 창이 시작되어야 하는 날짜와 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않으면 기본적으로 24시간 전으로 설정됩니다. 지정할 수 있는 가장 빠른 날짜는 7일 전입니다.

     YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. 다음을 채운 후 `trident-protect-support-bundle.yaml` 올바른 값으로 파일을 만들고 CR을 적용합니다.

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

CLI를 사용하여 지원 번들을 만듭니다.

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 지원 번들을 만듭니다. 그만큼 `trigger-type` 번들이 즉시

생성되는지 또는 생성 시간이 일정에 따라 결정되는지 여부를 결정하고 다음을 수행할 수 있습니다. `Manual` 또는 `Scheduled` . 기본 설정은 다음과 같습니다. `Manual` .

예를 들어:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

지원 번들을 모니터링하고 검색합니다.

두 가지 방법 중 하나를 사용하여 지원 번들을 만든 후에는 생성 진행 상황을 모니터링하고 로컬 시스템으로 검색할 수 있습니다.

단계

1. 기다리다 `status.generationState` 도달하다 `Completed` 상태. 다음 명령을 사용하여 생성 진행 상황을 모니터링할 수 있습니다.

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
```

2. 지원 번들을 로컬 시스템으로 가져옵니다. 완성된 AutoSupport 번들에서 복사 명령을 가져옵니다.

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

찾아라 `kubectl cp` 출력에서 명령을 선택하고 대상 인수를 원하는 로컬 디렉토리로 바꿔서 실행합니다.

Trident 프로텍트 업그레이드

Trident Protect를 최신 버전으로 업그레이드하면 새로운 기능이나 버그 수정을 활용할 수 있습니다.

> 버전 24.10에서 업그레이드하는 경우 업그레이드 중에 실행 중인 스냅샷이 실패할 수 있습니다. 이 오류로 인해 향후 스냅샷(수동 또는 예약)이 생성되는 것은 방해받지 않습니다. 업그레이드 중에 스냅샷이 실패하면 수동으로 새 스냅샷을 만들어서 애플리케이션을 보호할 수 있습니다.
>
> 잠재적인 실패를 방지하려면 업그레이드 전에 모든 스냅샷 일정을 비활성화한 다음 나중에 다시 활성화할 수 있습니다. 하지만 이로 인해 업그레이드 기간 동안 예약된 스냅샷이 누락되는 문제가 발생합니다.

Trident Protect를 업그레이드하려면 다음 단계를 수행하세요.

단계

1. Trident Helm 저장소를 업데이트하세요:

```
helm repo update
```

2. Trident Protect CRD 업그레이드:

> 25.06 이전 버전에서 업그레이드하는 경우 CRD가 이제 Trident Protect Helm 차트에 포함되었으므로 이 단계가 필요합니다.

a. CRD 관리를 전환하려면 이 명령을 실행하세요. `trident-protect-crds` 에게 `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Helm 비밀을 삭제하려면 이 명령을 실행하세요. `trident-protect-crds` 차트:

> 제거하지 마십시오 `trident-protect-crds` Helm을 사용하여 차트를 만들면 CRD와 관련 데이터가 제거될 수 있습니다.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident 프로텍트 업그레이드:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2506.0 --namespace trident-protect
```

애플리케이션 관리 및 보호

Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다**.**

Trident Protect의 버킷 사용자 정의 리소스(CR)는 AppVault로 알려져 있습니다. AppVault 객체는 스토리지 버킷의 선언적 Kubernetes 워크플로 표현입니다. AppVault CR에는 백업, 스냅샷, 복원 작업, SnapMirror 복제와 같은 보호 작업에 버킷을 사용하는 데 필요한 구성이 포함되어 있습니다. 관리자만 AppVault를 만들 수 있습니다.

애플리케이션에서 데이터 보호 작업을 수행할 때는 AppVault CR을 수동으로 또는 명령줄에서 생성해야 합니다. AppVault CR은 사용자 환경에 따라 다르므로, 이 페이지의 예시를 참고하여 AppVault CR을 생성할 수 있습니다.

Trident Protect가 설치된 클러스터에 AppVault CR이 있는지 확인하세요. AppVault CR이 없거나 액세스할 수 없는 경우 명령줄에 오류가 표시됩니다.

AppVault 인증 및 비밀번호 구성

AppVault CR을 생성하기 전에 AppVault와 선택한 데이터 이동자가 공급자 및 관련 리소스에 대해 인증할 수 있는지 확인하세요.

데이터 무버 저장소 비밀번호

CR이나 Trident Protect CLI 플러그인을 사용하여 AppVault 객체를 생성하는 경우 Restic 및 Kopia 암호화를 위한 사용자 정의 비밀번호가 포함된 Kubernetes 비밀번호를 지정할 수 있습니다. 비밀번호를 지정하지 않으면 Trident Protect는 기본 비밀번호를 사용합니다.

- AppVault CR을 수동으로 생성할 때 **spec.dataMoverPasswordSecretRef** 필드를 사용하여 비밀번호를 지정합니다.
- Trident Protect CLI를 사용하여 AppVault 객체를 생성할 때 다음을 사용하십시오. `--data-mover-password-secret-ref` 비밀을 지정하는 인수입니다.

데이터 무버 저장소 비밀번호 비밀을 만듭니다**.**

다음 예를 사용하여 비밀번호를 생성하세요. AppVault 객체를 생성할 때 Trident Protect가 이 비밀번호를 사용하여 데이터 이동 저장소를 인증하도록 지시할 수 있습니다.

- 사용하는 데이터 이동 서비스에 따라 해당 데이터 이동 서비스에 해당하는 비밀번호만 포함하면 됩니다. 예를 들어, Restic을 사용하고 있고 앞으로 Kopia를 사용할 계획이 없다면 비밀번호를 생성할 때 Restic 비밀번호만 포함할 수 있습니다.
- 비밀번호를 안전한 곳에 보관하세요. 같은 클러스터나 다른 클러스터에서 데이터를 복원할 때 필요합니다. 클러스터 또는 `trident-protect` 네임스페이스가 삭제되면 비밀번호 없이는 백업이나 스냅샷을 복원할 수 없습니다.

CR을 사용하세요

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

CLI를 사용하세요

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

S3 호환 스토리지 IAM 권한

Amazon S3, Generic S3 등 S3 호환 스토리지에 접속할 경우 "스토리지그리드 S3" , 또는 "ONTAP S3" Trident Protect를 사용하는 경우, 제공하는 사용자 자격 증명에 버킷에 액세스하는 데 필요한 권한이 있는지 확인해야 합니다. 다음은 Trident Protect에 대한 액세스에 필요한 최소한의 권한을 부여하는 정책의 예입니다. 이 정책은 S3 호환 버킷 정책을 관리하는 사용자에게 적용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon S3 정책에 대한 자세한 내용은 다음 예제를 참조하세요. "Amazon S3 문서" .

Amazon S3(AWS) 인증을 위한 EKS Pod ID

Trident Protect는 Kopia 데이터 이동 작업을 위해 EKS Pod Identity를 지원합니다. 이 기능을 사용하면 Kubernetes 비밀에 AWS 자격 증명을 저장하지 않고도 S3 버킷에 안전하게 액세스할 수 있습니다.

- Trident Protect를 사용한 EKS Pod Identity 요구 사항*

Trident Protect와 함께 EKS Pod Identity를 사용하기 전에 다음 사항을 확인하세요.

- EKS 클러스터에 Pod Identity가 활성화되어 있습니다.
- 필요한 S3 버킷 권한이 있는 IAM 역할을 생성했습니다. 자세한 내용은 다음을 참조하세요. "S3 호환 스토리지 IAM 권한".
- IAM 역할은 다음 Trident Protect 서비스 계정과 연결됩니다.
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Pod Identity를 활성화하고 IAM 역할을 서비스 계정과 연결하는 방법에 대한 자세한 지침은 다음을 참조하세요. "AWS EKS Pod Identity 문서" .

AppVault 구성 EKS Pod Identity를 사용하는 경우 AppVault CR을 다음과 같이 구성합니다. `useIAM: true` 명시적 자격 증명 대신 플래그:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

클라우드 공급자를 위한 AppVault 키 생성 예시

AppVault CR을 정의할 때 IAM 인증을 사용하지 않는 한 공급자가 호스팅하는 리소스에 액세스하기 위한 자격 증명을 포함해야 합니다. 자격 증명에 대한 키를 생성하는 방법은 공급자에 따라 다릅니다. 다음은 여러 공급자에 대한 명령줄 키 생성 예입니다. 다음 예를 사용하여 각 클라우드 공급자의 자격 증명에 대한 키를 생성할 수 있습니다.

구글 클라우드

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

아마존 **S3(AWS)**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

마이크로소프트 애저

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

일반 **S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

스토리지그리드 **S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

AppVault 생성 예시

각 공급자에 대한 AppVault 정의의 예는 다음과 같습니다.

AppVault CR 예시

다음 CR 예제를 사용하여 각 클라우드 공급자에 대한 AppVault 객체를 만들 수 있습니다.

- Restic 및 Kopia 저장소 암호화를 위한 사용자 정의 비밀번호가 포함된 Kubernetes 비밀번호를 선택적으로 지정할 수 있습니다. 참조하다데이터 무버 저장소 비밀번호 자세한 내용은.
- Amazon S3(AWS) AppVault 개체의 경우 선택적으로 sessionToken을 지정할 수 있습니다. 이는 인증을 위해 SSO(Single Sign-On)를 사용하는 경우에 유용합니다. 이 토큰은 공급자에 대한 키를 생성할 때 생성됩니다.클라우드 공급자를 위한 AppVault 키 생성 예시 .
- S3 AppVault 개체의 경우 선택적으로 다음을 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 있습니다. `spec.providerConfig.S3.proxyURL` 열쇠.

구글 클라우드

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

아마존 **S3(AWS)**

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

Kopia 데이터 무버와 함께 Pod Identity를 사용하는 EKS 환경의 경우 다음을 제거할 수 있습니다. `providerCredentials` 섹션을 추가하고 추가하세요 `useIAM: true` 아래에 `s3` 대신 구성을 사용하세요.

마이크로소프트 애저

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

일반 **S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

스토리지그리드 **S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Trident Protect CLI를 사용한 **AppVault** 생성 예

다음 CLI 명령 예를 사용하여 각 공급자에 대한 AppVault CR을 만들 수 있습니다.

- Restic 및 Kopia 저장소 암호화를 위한 사용자 정의 비밀번호가 포함된 Kubernetes 비밀번호를 선택적으로 지정할 수 있습니다. 참조하다데이터 무버 저장소 비밀번호 자세한 내용은.
- S3 AppVault 개체의 경우 선택적으로 다음을 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 있습니다. `--proxy-url <ip_address:port>` 논쟁.

구글 클라우드

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

아마존 **S3(AWS)**

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

마이크로소프트 애저

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

일반 **S3**

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

스토리지그리드 **S3**

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

AppVault 정보 보기

Trident Protect CLI 플러그인을 사용하면 클러스터에서 생성한 AppVault 개체에 대한 정보를 볼 수 있습니다.

단계

1. AppVault 개체의 내용을 봅니다.

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

예시 출력:

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+-----------------------------
+---------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+-----------------------------
+---------------------------+
```

2. 선택적으로 각 리소스에 대한 AppVaultPath를 보려면 플래그를 사용하세요. `--show-paths` .

   표의 첫 번째 열에 있는 클러스터 이름은 Trident Protect helm 설치 시 클러스터 이름이 지정된 경우에만 사용할 수 있습니다. 예를 들어: `--set clusterName=production1` .

AppVault 제거

언제든지 AppVault 객체를 제거할 수 있습니다.

> 제거하지 마십시오 `finalizers` AppVault 개체를 삭제하기 전에 AppVault CR에 키를 입력하세요. 그렇게 하면 AppVault 버킷에 잔여 데이터가 생기고 클러스터에 버려진 리소스가 생길 수 있습니다.

시작하기 전에

삭제하려는 AppVault에서 사용 중인 모든 스냅샷 및 백업 CR을 삭제했는지 확인하세요.

Kubernetes CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거하고 교체합니다. `appvault-name` 제거할 AppVault 개체의 이름:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거하고 교체합니다. `appvault-name` 제거할 AppVault 개체의 이름:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect를 사용하여 관리를 위한 애플리케이션 정의

Trident Protect로 관리하려는 애플리케이션을 정의하려면 애플리케이션 CR과 관련 AppVault CR을 생성하면 됩니다.

AppVault CR 만들기

애플리케이션에서 데이터 보호 작업을 수행할 때 사용할 AppVault CR을 만들어야 하며, AppVault CR은 Trident Protect가 설치된 클러스터에 있어야 합니다. AppVault CR은 사용자 환경에 따라 다릅니다. AppVault CR의 예는 다음을 참조하세요."AppVault 사용자 정의 리소스."

응용 프로그램 정의

Trident Protect로 관리하려는 각 애플리케이션을 정의해야 합니다. 수동으로 애플리케이션 CR을 생성하거나 Trident Protect CLI를 사용하여 관리할 애플리케이션을 정의할 수 있습니다.

CR을 사용하여 애플리케이션 추가

단계

1. 대상 애플리케이션 CR 파일을 만듭니다.
   a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `maria-app.yaml`).
   b. 다음 속성을 구성합니다.
      - **metadata.name**: (필수) 애플리케이션 사용자 정의 리소스의 이름입니다. 보호 작업에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 이름을 기록해 두세요.
      - **spec.includedNamespaces**: (필수) 네임스페이스와 레이블 선택기를 사용하여 애플리케이션이 사용하는 네임스페이스와 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 목록에 포함되어야 합니다. 레이블 선택기는 선택 사항이며 지정된 각 네임스페이스 내의 리소스를 필터링하는 데 사용할 수 있습니다.
      - **spec.includedClusterScopedResources**: (선택 사항) 이 특성을 사용하여 애플리케이션 정의에 포함될 클러스터 범위 리소스를 지정합니다. 이 속성을 사용하면 그룹, 버전, 종류 및 레이블을 기준으로 리소스를 선택할 수 있습니다.
        - **groupVersionKind**: (필수) 클러스터 범위 리소스의 API 그룹, 버전 및 종류를 지정합니다.
        - **labelSelector**: (선택 사항) 레이블을 기준으로 클러스터 범위 리소스를 필터링합니다.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (선택 사항) 이 주석은 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 정지되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 스냅샷 중에 이 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정합니다. true로 설정하면 애플리케이션은 글로벌 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. false로 설정하면 애플리케이션은 글로벌 설정을 무시하고 스냅샷을 찍는 동안 파일 시스템이 동결됩니다. 지정되었지만 애플리케이션 정의에 가상 머신이 없는 경우 주석은 무시됩니다. 지정하지 않으면 응용 프로그램은 다음을 따릅니다."글로벌 Trident 프로텍트 동결 설정" .

애플리케이션이 이미 생성된 후에 이 주석을 적용해야 하는 경우 다음 명령을 사용할 수 있습니다.

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
YAML 예시:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (선택 사항) 특정 레이블이 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

    - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `Include` 또는 `Exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.

        - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
            - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
            - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
            - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
            - **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.

- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

> ⓘ 둘 다 `resourceFilter` 그리고 `labelSelector` 사용됩니다, `resourceFilter` 먼저 실행한 다음 `labelSelector` 결과 리소스에 적용됩니다.

예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

2. 환경에 맞게 애플리케이션 CR을 만든 후 CR을 적용합니다. 예를 들어:

```
kubectl apply -f maria-app.yaml
```

단계

1. 다음 예제 중 하나를 사용하여 애플리케이션 정의를 만들고 적용하고, 괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예제에 표시된 인수와 함께 쉼표로 구분된 목록을 사용하여 애플리케이션 정의에 네임스페이스와 리소스를 포함할 수 있습니다.

   스냅샷 중에 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정하기 위해 앱을 생성할 때 주석을 사용할 수 있습니다. 이는 스냅샷 전에 파일 시스템이 정지되는 KubeVirt 환경과 같은 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 주석을 다음과 같이 설정하면 `true` , 애플리케이션은 글로벌 설정을 무시하고 스냅샷 동안 파일 시스템에 쓸 수 있습니다. 이것을 설정하면 `false` , 애플리케이션은 글로벌 설정을 무시하고 스냅샷을 찍는 동안 파일 시스템이 동결됩니다. 주석을 사용하지만 애플리케이션 정의에 가상 머신이 없는 경우 주석은 무시됩니다. 주석을 사용하지 않으면 응용 프로그램은 다음을 따릅니다."글로벌 Trident 프로텍트 동결 설정" .

CLI를 사용하여 애플리케이션을 생성할 때 주석을 지정하려면 다음을 사용할 수 있습니다. `--annotation` 깃발.

- 애플리케이션을 만들고 파일 시스템 동결 동작에 대한 글로벌 설정을 사용합니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace>
```

- 애플리케이션을 생성하고 파일 시스템 정지 동작에 대한 로컬 애플리케이션 설정을 구성합니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

사용할 수 있습니다 `--resource-filter-include` 그리고 `--resource-filter-exclude` 리소스를 포함하거나 제외하기 위한 플래그 `resourceSelectionCriteria` 다음 예에서 볼 수 있듯이 그룹, 종류, 버전, 레이블, 이름 및 네임스페이스와 같은 것들이 있습니다.

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
--resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-
namespace"],"LabelSelectors":["app=my-app"]}]'
```

Trident Protect를 사용하여 애플리케이션 보호

자동화된 보호 정책이나 임시 기반으로 스냅샷과 백업을 수행하여 Trident Protect가 관리하는 모든 앱을 보호할 수 있습니다.

데이터 보호 작업 중에 파일 시스템을 동결 및 동결 해제하도록 Trident Protect를 구성할 수 있습니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".

주문형 스냅샷 만들기

언제든지 주문형 스냅샷을 만들 수 있습니다.

클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 복제본에 포함됩니다.

CR을 사용하여 스냅샷을 만듭니다.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-snapshot-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠(메타데이터)를 저장해야 하는 AppVault의 이름입니다.
   - **spec.reclaimPolicy**: (선택 사항) 스냅샷 CR이 삭제될 때 스냅샷의 AppArchive에 어떤 일이 일어나는지 정의합니다. 이는 설정된 경우에도 다음을 의미합니다. `Retain` , 스냅샷이 삭제됩니다. 유효한 옵션:
     - `Retain`(기본)
     - `Delete`

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. 다음을 채운 후 `trident-protect-snapshot-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

CLI를 사용하여 스냅샷을 만듭니다.

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔 스냅샷을 만듭니다. 예를 들어:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

주문형 백업 만들기

언제든지 앱을 백업할 수 있습니다.

> 클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 복제본에 포함됩니다.

시작하기 전에

장기 실행 S3 백업 작업에 대해 AWS 세션 토큰 만료 기간이 충분한지 확인하세요. 백업 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
- 를 참조하세요 "AWS IAM 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

CR을 사용하여 백업을 만듭니다**.**

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-backup-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 내용을 저장해야 하는 AppVault의 이름입니다.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia`(기본)
   - **spec.reclaimPolicy**: (선택 사항) 백업이 클레임에서 해제될 때 발생하는 작업을 정의합니다. 가능한 값:
     - `Delete`
     - `Retain`(기본)
   - **spec.snapshotRef**: (선택 사항): 백업 소스로 사용할 스냅샷의 이름입니다. 제공되지 않으면 임시 스냅샷이 생성되어 백업됩니다.
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (선택 사항) 이 주석은 백업이 증분 방식이 아니어야 하는지 여부를 지정하는 데 사용됩니다. 기본적으로 모든 백업은 증분 백업입니다. 그러나 이 주석이 설정된 경우 `true` , 백업이 비증분 백업으로 전환됩니다. 지정하지 않으면 백업은 기본 증분 백업 설정을 따릅니다. 복원과 관련된 위험을 최소화하려면 주기적으로 전체 백업을 수행한 다음 전체 백업 사이에 증분 백업을 수행하는 것이 가장 좋습니다.

     YAML 예시:

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup: "true"
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. 다음을 채운 후 `trident-protect-backup-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-backup-cr.yaml
```

CLI를 사용하여 백업을 만듭니다.

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔 백업을 만듭니다. 예를 들어:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

선택적으로 사용할 수 있습니다 `--full-backup` 백업이 증분되지 않아야 하는지 여부를 지정하는 플래그입니다. 기본적으로 모든 백업은 증분 백업입니다. 이 플래그를 사용하면 백업이 비증분 방식으로 진행됩니다. 복원과 관련된 위험을 최소화하려면 주기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 가장 좋습니다.

데이터 보호 일정을 만듭니다

보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 매시간, 매일, 매주, 매월 스냅샷과 백업을 만들도록 선택할 수 있으며, 보관할 복사본 수를 지정할 수 있습니다. full-backup-rule 주석을 사용하여 비증분 전체 백업을 예약할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 주기적으로 전체 백업을 수행하고 그 사이에 증분 백업을 수행하면 복원과 관련된 위험을 줄이는 데 도움이 됩니다.

- 스냅샷에 대한 일정은 다음을 설정하여 생성할 수 있습니다. `backupRetention` 0으로 그리고 `snapshotRetention` 0보다 큰 값으로. 환경 `snapshotRetention` 0으로 설정하면 예약된 백업은 여전히 스냅샷을 생성하지만, 이는 일시적이며 백업이 완료되면 즉시 삭제됩니다.
- 클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 복제본에 포함됩니다.

CR을 사용하여 일정을 만듭니다**.**

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-schedule-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.

   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia`(기본)
   - **spec.applicationRef**: 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 내용을 저장해야 하는 AppVault의 이름입니다.
   - **spec.backupRetention**: 보관할 백업 수. 0은 백업을 생성하지 않음을 나타냅니다(스냅샷만 생성).
   - **spec.snapshotRetention**: 보관할 스냅샷 수. 0은 스냅샷을 생성하지 않음을 나타냅니다.
   - **spec.granularity**: 일정을 실행해야 하는 빈도입니다. 가능한 값과 필수 연관 필드는 다음과 같습니다.
     - `Hourly`(지정해야 함) `spec.minute`)
     - `Daily`(지정해야 함) `spec.minute` 그리고 `spec.hour`)
     - `Weekly`(지정해야 함) `spec.minute, spec.hour` , 그리고 `spec.dayOfWeek`)
     - `Monthly`(지정해야 함) `spec.minute, spec.hour` , 그리고 `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (선택 사항) 일정을 실행해야 하는 날짜(1~31)입니다. 이 필드는 세분성이 설정된 경우 필수입니다. `Monthly` . 값은 문자열로 제공되어야 합니다.
   - **spec.dayOfWeek**: (선택 사항) 일정을 실행해야 하는 요일(0~7). 0 또는 7의 값은 일요일을 나타냅니다. 이 필드는 세분성이 설정된 경우 필수입니다. `Weekly` . 값은 문자열로 제공되어야 합니다.
   - **spec.hour**: (선택 사항) 일정을 실행해야 하는 시간(0~23)입니다. 이 필드는 세분성이 설정된 경우 필수입니다. `Daily` , `Weekly` , 또는 `Monthly` . 값은 문자열로 제공되어야 합니다.
   - **spec.minute**: (선택 사항) 일정을 실행해야 하는 분(0~59)입니다. 이 필드는 세분성이 설정된 경우 필수입니다. `Hourly` , `Daily` , `Weekly` , 또는 `Monthly` . 값은 문자열로 제공되어야 합니다.
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (선택 사항) 이 주석은 전체 백업을 예약하기 위한 규칙을 지정하는 데 사용됩니다. 설정할 수 있습니다 `always` 지속적인 전체 백업을 위해 사용하거나 요구 사항에 맞게 사용자 정의할 수 있습니다. 예를 들어, 일별 단위를 선택하면 전체 백업을 수행할 요일을 지정할 수 있습니다.

     백업 및 스냅샷 일정에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
  annotations:
    protect.trident.netapp.io/full-backup-rule: "Monday,Thursday"
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

스냅샷 전용 일정에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

3. 다음을 채운 후 `trident-protect-schedule-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

CLI를 사용하여 일정을 만듭니다.

단계

1. 보호 일정을 만들고 괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예를 들어:

   사용할 수 있습니다 `tridentctl-protect create schedule --help` 이 명령에 대한 자세한 도움말 정보를 보려면.

   ```
   tridentctl-protect create schedule <my_schedule_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> --backup
   -retention <how_many_backups_to_retain> --data-mover
   <Kopia_or_Restic> --day-of-month <day_of_month_to_run_schedule>
   --day-of-week <day_of_month_to_run_schedule> --granularity
   <frequency_to_run> --hour <hour_of_day_to_run> --minute
   <minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
   -retention <how_many_snapshots_to_retain> -n <application_namespace>
   --full-backup-rule <string>
   ```

   설정할 수 있습니다 `--full-backup-rule` 플래그를 `always` 지속적인 전체 백업을 위해 사용하거나 요구 사항에 맞게 사용자 정의할 수 있습니다. 예를 들어, 일 단위를 선택하면 전체 백업을 수행할 요일을 지정할 수 있습니다. 예를 들어, 사용 `--full-backup-rule "Monday,Thursday"` 월요일과 목요일에 전체 백업을 예약합니다.

   스냅샷 전용 일정의 경우 다음을 설정합니다. `--backup-retention 0` 0보다 큰 값을 지정합니다. `--snapshot-retention` .

스냅샷 삭제

더 이상 필요하지 않은 예약된 스냅샷이나 주문형 스냅샷을 삭제합니다.

단계

1. 스냅샷과 연관된 스냅샷 CR을 제거합니다.

   ```
   kubectl delete snapshot <snapshot_name> -n my-app-namespace
   ```

백업 삭제

더 이상 필요하지 않은 예약된 백업이나 주문형 백업을 삭제합니다.

회수 정책이 설정되어 있는지 확인하십시오. `Delete` 개체 스토리지에서 모든 백업 데이터를 제거합니다. 정책의 기본 설정은 다음과 같습니다. `Retain` 실수로 데이터가 손실되는 것을 방지합니다. 정책이 변경되지 않으면 `Delete` 백업 데이터는 개체 스토리지에 남아 있으므로 수동으로 삭제해야 합니다.

단계

1. 백업과 관련된 백업 CR을 제거합니다.

```
kubectl delete backup <backup_name> -n my-app-namespace
```

백업 작업 상태 확인

명령줄을 사용하여 진행 중인 백업 작업, 완료된 백업 작업 또는 실패한 백업 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 백업 작업의 상태를 검색하고, 괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

azure-netapp-files(ANF) 작업에 대한 백업 및 복원 활성화

Trident Protect를 설치한 경우 azure-netapp-files 스토리지 클래스를 사용하고 Trident 24.06 이전에 생성된 스토리지 백엔드에 대해 공간 효율적인 백업 및 복원 기능을 활성화할 수 있습니다. 이 기능은 NFSv4 볼륨에서 작동하며 용량 풀에서 추가 공간을 소모하지 않습니다.

시작하기 전에

다음 사항을 확인하세요.

- Trident Protect를 설치했습니다.
- Trident Protect에서 애플리케이션을 정의했습니다. 이 절차를 완료하기 전까지 이 애플리케이션의 보호 기능은 제한됩니다.
- 당신은 가지고있다 `azure-netapp-files` 스토리지 백엔드의 기본 스토리지 클래스로 선택되었습니다.

구성 단계를 확장하세요

1. Trident 24.10으로 업그레이드하기 전에 ANF 볼륨이 생성된 경우 Trident 에서 다음을 수행하세요.

   a. azure-netapp-files 기반이며 애플리케이션과 연결된 각 PV에 대해 스냅샷 디렉토리를 활성화합니다.

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. 연관된 각 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인하세요.

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   응답:

   ```
   snapshotDirectory: "true"
   ```

   +
   스냅샷 디렉토리가 활성화되어 있지 않으면 정기적인 백업 기능을 선택합니다. 이 기능은 백업 프로세스 중에 용량 풀의 공간을 일시적으로 사용합니다. 이 경우 백업되는 볼륨 크기의 임시 볼륨을 생성할 수 있는 충분한 공간이 용량 풀에 있는지 확인하세요.

결과

Trident Protect를 사용하여 애플리케이션을 백업하고 복원할 준비가 되었습니다. 각 PVC는 다른 애플리케이션에서 백업 및 복원을 위해 사용할 수도 있습니다.

응용 프로그램 복원

Trident Protect를 사용하여 애플리케이션 복원

Trident Protect를 사용하면 스냅샷이나 백업에서 애플리케이션을 복원할 수 있습니다. 동일한 클러스터로 애플리케이션을 복원할 때 기존 스냅샷에서 복원하는 것이 더 빠릅니다.

- 애플리케이션을 복원하면 해당 애플리케이션에 구성된 모든 실행 후크가 앱과 함께 복원됩니다. 복원 후 실행 후크가 있는 경우 복원 작업의 일부로 자동으로 실행됩니다.
- qtree 볼륨의 경우 백업에서 다른 네임스페이스나 원래 네임스페이스로 복원하는 기능이 지원됩니다. 그러나 qtree 볼륨의 경우 스냅샷에서 다른 네임스페이스나 원래 네임스페이스로 복원하는 작업은 지원되지 않습니다.
- 고급 설정을 사용하여 복원 작업을 사용자 정의할 수 있습니다. 자세한 내용은 다음을 참조하세요. "고급 Trident Protect 복원 설정 사용".

BackupRestore CR을 사용하여 다른 네임스페이스로 백업을 복원하면 Trident Protect는 새 네임스페이스에서 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 만듭니다. 복구된 애플리케이션을 보호하려면 주문형 백업이나 스냅샷을 만들거나 보호 일정을 설정하세요.

기존 리소스가 있는 다른 네임스페이스에 백업을 복원하더라도 백업에 있는 리소스와 이름을 공유하는 리소스는 변경되지 않습니다. 백업에 있는 모든 리소스를 복원하려면 대상 네임스페이스를 삭제하고 다시 만들거나 백업을 새 네임스페이스로 복원합니다.

시작하기 전에

AWS 세션 토큰 만료일이 장기 실행 s3 복원 작업에 충분한지 확인하세요. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
- 를 참조하세요 "AWS IAM 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-backup-restore-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (필수) 백업 내용이 저장된 AppVault의 이름입니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 대상 네임스페이스로 매핑합니다. 바꾸다 `my-source-namespace` 그리고 `my-destination-namespace` 주변 환경의 정보를 활용하여

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (선택 사항) 복원할 애플리케이션의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

   > Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `Include` 또는 `Exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.

- **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
- **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
- **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
- **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

  예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 다음을 채운 후 `trident-protect-backup-restore-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI를 사용하세요

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 백업을 다른 네임스페이스로 복원합니다. 그만큼 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 다음 형식의 올바른 대상 네임스페이스에 매핑합니다. `source1:dest1,source2:dest2` . 예를 들어:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

백업에서 원래 네임스페이스로 복원

언제든지 원래 네임스페이스로 백업을 복원할 수 있습니다.

시작하기 전에

AWS 세션 토큰 만료일이 장기 실행 s3 복원 작업에 충분한지 확인하세요. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
- 를 참조하세요 "AWS IAM 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-backup-ipr-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (필수) 백업 내용이 저장된 AppVault의 이름입니다.

     예를 들어:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (선택 사항) 복원할 애플리케이션의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

   Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `Include` 또는 `Exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.

- **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
- **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

  예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 다음을 채운 후 `trident-protect-backup-ipr-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

CLI를 사용하세요

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 백업을 원래 네임스페이스로 복원합니다. 그만큼 `backup` 인수는 형식에서 네임스페이스와 백업 이름을 사용합니다. `<namespace>/<name>` . 예를 들어:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

원래 클러스터에 문제가 있는 경우 다른 클러스터로 백업을 복원할 수 있습니다.

Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

시작하기 전에

다음 전제 조건이 충족되는지 확인하세요.

- 대상 클러스터에 Trident Protect가 설치되어 있습니다.
- 대상 클러스터는 백업이 저장된 소스 클러스터와 동일한 AppVault의 버킷 경로에 액세스할 수 있습니다.
- AppVault CR에서 정의된 개체 저장소 버킷에 로컬 환경이 연결할 수 있는지 확인하십시오. `tridentctl-protect get appvaultcontent` 명령. 네트워크 제한으로 인해 액세스가 불가능한 경우 대상 클러스터의 포드 내에서 Trident Protect CLI를 실행하세요.
- AWS 세션 토큰 만료일이 장기 복원 작업에 충분한지 확인하세요. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.
  - 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
  - 를 참조하세요 "AWS 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

단계

1. Trident Protect CLI 플러그인을 사용하여 대상 클러스터에서 AppVault CR의 가용성을 확인하세요.

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

애플리케이션 복원에 사용되는 네임스페이스가 대상 클러스터에 있는지 확인하세요.

2. 대상 클러스터에서 사용 가능한 AppVault의 백업 내용을 확인합니다.

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

이 명령을 실행하면 AppVault에서 사용 가능한 백업이 표시됩니다. 여기에는 원래 클러스터, 해당 애플리케이션 이름, 타임스탬프, 보관 경로가 포함됩니다.

예시 출력:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. AppVault 이름과 보관 경로를 사용하여 대상 클러스터에 애플리케이션을 복원합니다.

CR을 사용하세요

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-backup-restore-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 백업 내용이 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 백업 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     BackupRestore CR을 사용할 수 없는 경우 2단계에서 언급한 명령을 사용하여 백업 내용을 볼 수 있습니다.

   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 대상 네임스페이스로 매핑합니다. 바꾸다 `my-source-namespace` 그리고 `my-destination-namespace` 주변 환경의 정보를 활용하여

     예를 들어:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 다음을 채운 후 `trident-protect-backup-restore-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI를 사용하세요

1. 다음 명령을 사용하여 애플리케이션을 복원하고 괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 네임스페이스 매핑 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 source1:dest1,source2:dest2

형식으로 올바른 대상 네임스페이스에 매핑합니다. 예를 들어:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

스냅샷에서 다른 네임스페이스로 복원

사용자 정의 리소스(CR) 파일을 사용하여 스냅샷에서 데이터를 다른 네임스페이스나 원래 소스 네임스페이스로 복원할 수 있습니다. SnapshotRestore CR을 사용하여 다른 네임스페이스로 스냅샷을 복원하면 Trident Protect는 새 네임스페이스에서 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 만듭니다. 복구된 애플리케이션을 보호하려면 주문형 백업이나 스냅샷을 만들거나 보호 일정을 설정하세요.

SnapshotRestore는 다음을 지원합니다. `spec.storageClassMapping` 속성이지만 소스 및 대상 저장소 클래스가 동일한 저장소 백엔드를 사용하는 경우에만 해당됩니다. 복원을 시도하는 경우 `StorageClass` 다른 스토리지 백엔드를 사용하는 경우 복원 작업이 실패합니다.

시작하기 전에

AWS 세션 토큰 만료일이 장기 실행 s3 복원 작업에 충분한지 확인하세요. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
- 를 참조하세요 "AWS IAM 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-snapshot-restore-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 내용이 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 대상 네임스페이스로 매핑합니다. 바꾸다 `my-source-namespace` 그리고 `my-destination-namespace` 주변 환경의 정보를 활용하여

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (선택 사항) 복원할 애플리케이션의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

   > Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `Include` 또는 `Exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.

- **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
- **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
- **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
- **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

  예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 다음을 채운 후 `trident-protect-snapshot-restore-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI를 사용하세요

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 스냅샷을 다른 네임스페이스로 복원합니다.

   - 그만큼 `snapshot` 인수는 형식에 네임스페이스와 스냅샷 이름을 사용합니다. `<namespace>/<name>` .
   - 그만큼 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 형식에 맞는 올바른 대상 네임스페이스에 매핑합니다. `source1:dest1,source2:dest2` .

예를 들어:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

스냅샷에서 원래 네임스페이스로 복원

언제든지 원래 네임스페이스로 스냅샷을 복원할 수 있습니다.

시작하기 전에

AWS 세션 토큰 만료일이 장기 실행 s3 복원 작업에 충분한지 확인하세요. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 를 참조하세요 "AWS API 문서" 현재 세션 토큰 만료일을 확인하는 방법에 대한 자세한 내용은 다음을 참조하세요.
- 를 참조하세요 "AWS IAM 문서" AWS 리소스에 대한 자격 증명에 대한 자세한 내용은 다음을 참조하세요.

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-snapshot-ipr-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 내용이 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

```
kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
    appArchivePath: my-snapshot-path
```

3. (선택 사항) 복원할 애플리케이션의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

> ⓘ Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `Include` 또는 `Exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
       - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.
       - **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name

필드에 있는 이름입니다.

- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

  예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 다음을 채운 후 `trident-protect-snapshot-ipr-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용합니다.

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

CLI를 사용하세요

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 스냅샷을 원래 네임스페이스로 복원합니다. 예를 들어:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <snapshot_to_restore> \
-n <application_namespace>
```

복원 작업 상태 확인

명령줄을 사용하여 진행 중인 복원 작업, 완료된 복원 작업 또는 실패한 복원 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 복원 작업의 상태를 검색하고, 괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

고급 **Trident Protect** 복원 설정 사용

주석, 네임스페이스 설정, 스토리지 옵션 등의 고급 설정을 사용하여 복원 작업을 사용자 정의하여 특정 요구 사항을 충족할 수 있습니다.

복원 및 장애 조치 작업 중 네임스페이스 주석 및 레이블

복원 및 장애 조치 작업 동안 대상 네임스페이스의 레이블과 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 만들어집니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블이나 주석이 추가되고, 이미 존재하는 레이블이나 주석은 소스 네임스페이스의 값과 일치하도록 덮어씁니다. 대상 네임스페이스에만 존재하는 레이블이나 주석은 변경되지 않습니다.

> Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요. "OpenShift 보안 컨텍스트 제약 조건 문서" .

Kubernetes 환경 변수를 설정하여 대상 네임스페이스의 특정 주석이 덮어쓰여지는 것을 방지할 수 있습니다. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` 복원이나 장애 조치 작업을 수행하기 전에. 예를 들어:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

> 복원 또는 장애 조치 작업을 수행할 때 지정된 모든 네임스페이스 주석 및 레이블 `restoreSkipNamespaceAnnotations` 그리고 `restoreSkipNamespaceLabels` 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한 내용은 다음을 참조하세요. "AutoSupport 및 네임스페이스 필터링 옵션 구성".

Helm을 사용하여 소스 애플리케이션을 설치한 경우 `--create-namespace` 깃발에는 특별한 대우가 주어집니다. `name` 라벨 키. 복구 또는 장애 조치 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 사항 없이 대상 네임스페이스에 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태를 볼 수 있으며, 대상 네임스페이스에서 주석과 레이블이 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복구 또는 장애 조치 작업 전

다음 표는 복원 또는 장애 조치 작업 전의 예제 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "업데이트된 값" • annotation.two/key: "true"	• 환경=생산 • 규정 준수=HIPAA • 이름=ns-1
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• 역할=데이터베이스

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후의 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고 일부는 덮어쓰여졌으며 `name` 레이블이 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "업데이트된 값" • annotation.two/key: "true" • annotation.three/key: "false"	• 이름=ns-2 • 규정 준수=HIPAA • 환경=생산 • 역할=데이터베이스

지원되는 필드

이 섹션에서는 복원 작업에 사용할 수 있는 추가 필드에 대해 설명합니다.

스토리지 클래스 매핑

그만큼 `spec.storageClassMapping` 속성은 소스 애플리케이션에 있는 스토리지 클래스에서 대상 클러스터의 새 스토리지 클래스로의 매핑을 정의합니다. 서로 다른 스토리지 클래스를 사용하는 클러스터 간에 애플리케이션을 마이그레이션하거나 BackupRestore 작업에 대한 스토리지 백엔드를 변경할 때 이 기능을 사용할 수 있습니다.

예:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

지원되는 주석

이 섹션에서는 시스템의 다양한 동작을 구성하는 데 지원되는 주석을 나열합니다. 사용자가 주석을 명시적으로 설정하지 않으면 시스템은 기본값을 사용합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/데이터 이동 시간 초과 초	끈	데이터 이동 작업이 중단될 수 있는 최대 시간(초)입니다.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	끈	Kopia 콘텐츠 캐시의 최대 크기 제한(메가바이트)입니다.	"1000"

NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션 복제

Trident Protect를 사용하면 NetApp SnapMirror 기술의 비동기 복제 기능을 사용하여 동일한 클러스터 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 한 스토리지 백엔드에서 다른 스토리지 백엔드로 복제할 수 있습니다.

복원 및 장애 조치 작업 중 네임스페이스 주석 및 레이블

복원 및 장애 조치 작업 동안 대상 네임스페이스의 레이블과 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 만들어집니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블이나 주석이 추가되고, 이미 존재하는 레이블이나 주석은 소스 네임스페이스의 값과 일치하도록 덮어씁니다. 대상 네임스페이스에만 존재하는 레이블이나 주석은 변경되지 않습니다.

> Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요. "OpenShift 보안 컨텍스트 제약 조건 문서" .

Kubernetes 환경 변수를 설정하여 대상 네임스페이스의 특정 주석이 덮어쓰여지는 것을 방지할 수 있습니다. `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` 복원이나 장애 조치 작업을 수행하기 전에. 예를 들어:

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

복원 또는 장애 조치 작업을 수행할 때 지정된 모든 네임스페이스 주석 및 레이블 `restoreSkipNamespaceAnnotations` 그리고 `restoreSkipNamespaceLabels` 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한 내용은 다음을 참조하세요. "AutoSupport 및 네임스페이스 필터링 옵션 구성".

Helm을 사용하여 소스 애플리케이션을 설치한 경우 `--create-namespace` 깃발에는 특별한 대우가 주어집니다. `name` 라벨 키. 복구 또는 장애 조치 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 사항 없이 대상 네임스페이스에 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태를 볼 수 있으며, 대상 네임스페이스에서 주석과 레이블이 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복구 또는 장애 조치 작업 전

다음 표는 복원 또는 장애 조치 작업 전의 예제 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "업데이트된 값" • annotation.two/key: "true"	• 환경=생산 • 규정 준수=HIPAA • 이름=ns-1
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• 역할=데이터베이스

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후의 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고 일부는 덮어쓰여졌으며 `name` 레이블이 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "업데이트된 값" • annotation.two/key: "true" • annotation.three/key: "false"	• 이름=ns-2 • 규정 준수=HIPAA • 환경=생산 • 역할=데이터베이스

데이터 보호 작업 중에 파일 시스템을 동결 및 동결 해제하도록 Trident Protect를 구성할 수 있습니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".

장애 조치 및 역방향 작업 중 실행 후크

AppMirror 관계를 사용하여 애플리케이션을 보호하는 경우 장애 조치 및 역방향 작업 중에 알아야 할 실행 후크와 관련된 특정 동작이 있습니다.

- 장애 조치 중에 실행 후크는 소스 클러스터에서 대상 클러스터로 자동으로 복사됩니다. 수동으로 다시 만들 필요는 없습니다. 장애 조치 후에는 애플리케이션에 실행 후크가 존재하며 관련 작업 중에 실행됩니다.
- 역방향 또는 역방향 재동기화 중에 애플리케이션의 기존 실행 후크는 모두 제거됩니다. 소스 애플리케이션이 대상 애플리케이션이 되면 이러한 실행 후크는 유효하지 않으며 실행을 방지하기 위해 삭제됩니다.

실행 후크에 대해 자세히 알아보려면 다음을 참조하세요."Trident Protect 실행 후크 관리" .

복제 관계 설정

복제 관계를 설정하는 데는 다음이 포함됩니다.

- Trident Protect가 앱 스냅샷을 얼마나 자주 찍을지 선택합니다(여기에는 앱의 Kubernetes 리소스와 각 앱 볼륨에 대한 볼륨 스냅샷이 포함됨)
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 찍을 시간 설정

단계

1. 소스 클러스터에서 소스 애플리케이션에 대한 AppVault를 만듭니다. 저장소 공급자에 따라 다음 예제를 수정하세요."AppVault 사용자 정의 리소스" 귀하의 환경에 맞게:

CR을 사용하여 **AppVault** 만들기

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-appvault-primary-source.yaml`).

b. 다음 속성을 구성합니다.

- **metadata.name**: (필수) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 이름을 기록해 두세요.
- **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. 공급자에 대한 버킷 이름과 기타 필요한 세부 정보를 선택하세요. 복제 관계에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 값을 기록해 두세요. 참조하다"AppVault 사용자 정의 리소스" 다른 공급업체와의 AppVault CR 사례는 다음과 같습니다.
- **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
  - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀에서 나와야 함을 나타냅니다.
    - 키: (필수) 선택할 비밀번호의 유효한 키입니다.
    - **name**: (필수) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.
  - **spec.providerCredentials.secretAccessKey**: (필수) 공급자에 액세스하는 데 사용되는 액세스 키입니다. *name*은 *spec.providerCredentials.valueFromSecret.name*과 일치해야 합니다.
- **spec.providerType**: (필수) 백업을 제공하는 항목을 결정합니다. 예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure. 가능한 값:
  - AWS
  - 하늘빛
  - 지씨피
  - 제네릭-s3
  - 온탭-s3
  - 스토리지그리드-s3

c. 다음을 채운 후 `trident-protect-appvault-primary-source.yaml` 올바른 값으로 파일을 만들고 CR을 적용합니다.

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

CLI를 사용하여 **AppVault** 만들기

a. 괄호 안의 값을 사용자 환경의 정보로 바꿔 AppVault를 만듭니다.

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. 소스 클러스터에서 소스 애플리케이션 CR을 만듭니다.

CR을 사용하여 소스 애플리케이션을 만듭니다.

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-app-source.yaml`).

b. 다음 속성을 구성합니다.

- **metadata.name**: (필수) 애플리케이션 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 이름을 기록해 두세요.
- **spec.includedNamespaces**: (필수) 네임스페이스와 관련 레이블의 배열입니다. 네임스페이스 이름을 사용하고 필요에 따라 레이블을 사용하여 네임스페이스의 범위를 좁혀 여기에 나열된 네임스페이스에 있는 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 배열의 일부여야 합니다.

  YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. 다음을 채운 후 `trident-protect-app-source.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

CLI를 사용하여 소스 애플리케이션 만들기

a. 소스 애플리케이션을 생성합니다. 예를 들어:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. 선택적으로 소스 클러스터에서 소스 애플리케이션의 스냅샷을 만듭니다. 이 스냅샷은 대상 클러스터의 애플리케이션의 기반으로 사용됩니다. 이 단계를 건너뛰면 다음 예약된 스냅샷이 실행될 때까지 기다려야 최신 스냅샷을 얻을 수 있습니다. 주문형 스냅샷을 생성하려면 다음을 참조하세요. "주문형 스냅샷 만들기".
4. 소스 클러스터에서 복제 일정 CR을 만듭니다.

아래에 제공된 일정과 함께, 피어링된 ONTAP 클러스터 간에 공통 스냅샷을 유지하려면 7일의 보존 기간을 갖는 별도의 일일 스냅샷 일정을 만드는 것이 좋습니다. 이를 통해 스냅샷을 최대 7일 동안 사용할 수 있지만, 보존 기간은 사용자 요구 사항에 따라 사용자 정의할 수 있습니다.

장애 조치가 발생하면 시스템은 이러한 스냅샷을 최대 7일 동안 역방향 작업에 사용할 수 있습니다. 이 접근 방식을 사용하면 마지막 스냅샷 이후에 변경된 내용만 전송되고 모든 데이터는 전송되지 않으므로 역방향 프로세스가 더 빠르고 효율적입니다.

해당 애플리케이션의 기존 일정이 이미 원하는 보존 요구 사항을 충족하는 경우 추가 일정은 필요하지 않습니다.

CR을 사용하여 복제 일정을 만듭니다.

a. 소스 애플리케이션에 대한 복제 일정을 만듭니다.

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-schedule.yaml`).

ii. 다음 속성을 구성합니다.

- **metadata.name**: (필수) 일정 사용자 정의 리소스의 이름입니다.
- **spec.appVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.applicationRef**: (필수) 이 값은 소스 애플리케이션 CR의 metadata.name 필드와 일치해야 합니다.
- **spec.backupRetention**: (필수) 이 필드는 필수이며, 값은 0으로 설정해야 합니다.
- **spec.enabled**: true로 설정해야 합니다.
- **spec.granularity**: 설정해야 함 `Custom` .
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 정의합니다.
- **spec.snapshotRetention**: 2로 설정해야 합니다.

YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. 다음을 채운 후 `trident-protect-schedule.yaml` 올바른 값으로 파일을 만들고 CR을 적용합니다.

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

CLI를 사용하여 복제 일정을 만듭니다.

a. 괄호 안의 값을 사용자 환경의 정보로 바꿔 복제 일정을 만듭니다.

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

예:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. 대상 클러스터에서 소스 클러스터에 적용한 AppVault CR과 동일한 소스 애플리케이션 AppVault CR을 만들고 이름을 지정합니다(예: `trident-protect-appvault-primary-destination.yaml`).

6. CR을 적용하세요:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. 대상 클러스터의 대상 애플리케이션에 대한 대상 AppVault CR을 만듭니다. 저장소 공급자에 따라 다음 예제를 수정하세요."AppVault 사용자 정의 리소스" 귀하의 환경에 맞게:

   a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-appvault-secondary-destination.yaml`).

   b. 다음 속성을 구성합니다.

      - **metadata.name**: (필수) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 이름을 기록해 두세요.
      - **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. 선택하다 `bucketName` 그리고 귀하의 공급업체에 대한 기타 필요한 세부 정보입니다. 복제 관계에 필요한 다른 CR 파일이 이 값을 참조하므로 선택한 값을 기록해 두세요. 참조하다"AppVault 사용자 정의 리소스" 다른 공급업체와의 AppVault CR 사례는 다음과 같습니다.
      - **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
        - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀에서 나와야 함을 나타냅니다.
          - 키: (필수) 선택할 비밀번호의 유효한 키입니다.
          - **name**: (필수) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.

- **spec.providerCredentials.secretAccessKey**: (필수) 공급자에 액세스하는 데 사용되는 액세스 키입니다. *name*은 *spec.providerCredentials.valueFromSecret.name*과 일치해야 합니다.

- **spec.providerType**: (필수) 백업을 제공하는 항목을 결정합니다. 예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure. 가능한 값:
  - AWS
  - 하늘빛
  - 지씨피
  - 제네릭-s3
  - 온탭-s3
  - 스토리지그리드-s3

c. 다음을 채운 후 `trident-protect-appvault-secondary-destination.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. 대상 클러스터에서 AppMirrorRelationship CR 파일을 만듭니다.

CR을 사용하여 **AppMirrorRelationship**을 만듭니다**.**

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-relationship.yaml`).

b. 다음 속성을 구성합니다.

- **metadata.name:** (필수) AppMirrorRelationship 사용자 정의 리소스의 이름입니다.
- **spec.destinationAppVaultRef**: (필수) 이 값은 대상 클러스터의 대상 애플리케이션에 대한 AppVault 이름과 일치해야 합니다.
- **spec.namespaceMapping**: (필수) 대상 및 소스 네임스페이스는 해당 애플리케이션 CR에 정의된 애플리케이션 네임스페이스와 일치해야 합니다.
- **spec.sourceAppVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault 이름과 일치해야 합니다.
- **spec.sourceApplicationName**: (필수) 이 값은 소스 애플리케이션 CR에 정의한 소스 애플리케이션의 이름과 일치해야 합니다.
- **spec.sourceApplicationUID**: (필수) 이 값은 소스 애플리케이션 CR에서 정의한 소스 애플리케이션의 UID와 일치해야 합니다.
- **spec.storageClassName**: (선택 사항) 클러스터에서 유효한 스토리지 클래스의 이름을 선택합니다. 스토리지 클래스는 소스 환경과 피어링된 ONTAP 스토리지 VM에 연결되어야 합니다. 스토리지 클래스가 제공되지 않으면 기본적으로 클러스터의 기본 스토리지 클래스가 사용됩니다.
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 정의합니다.

YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. 다음을 채운 후 `trident-protect-relationship.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

CLI를 사용하여 **AppMirrorRelationship**을 만듭니다.

a. AppMirrorRelationship 객체를 만들고 적용하며, 괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

```
tridentctl-protect create appmirrorrelationship <name_of_appmirorrelationship> --destination-app-vault <my_vault_name> --source-app-vault <my_vault_name> --recurrence-rule <rule> --namespace-mapping <ns_mapping> --source-app-id <source_app_UID> --source-app <my_source_app_name> --storage-class <storage_class_name> -n <application_namespace>
```

예:

```
tridentctl-protect create appmirrorrelationship my-amr --destination-app-vault appvault2 --source-app-vault appvault1 --recurrence-rule "DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5" --source-app my-app --namespace-mapping "my-source-ns1:my-dest-ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-dest-ns1
```

9. (선택 사항) 대상 클러스터에서 복제 관계의 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath='{.status}' | jq
```

대상 클러스터로 장애 조치

Trident Protect를 사용하면 복제된 애플리케이션을 대상 클러스터로 장애 조치할 수 있습니다. 이 절차는 복제 관계를 중지하고 대상 클러스터에서 앱을 온라인으로 전환합니다. Trident Protect는 소스 클러스터에서 앱이 작동 중이면 앱을 중지하지 않습니다.

단계

1. 대상 클러스터에서 AppMirrorRelationship CR 파일을 편집합니다(예: `trident-protect-relationship.yaml`) 및 **spec.desiredState** 값을 다음으로 변경합니다. `Promoted` .

2. CR 파일을 저장합니다.
3. CR을 적용하세요:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (선택 사항) 장애 조치된 애플리케이션에 필요한 보호 일정을 만듭니다.
5. (선택 사항) 복제 관계의 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

장애가 발생한 복제 관계 재동기화

재동기화 작업은 복제 관계를 재설정합니다. 재동기화 작업을 수행한 후에는 원래 소스 애플리케이션이 실행 중인 애플리케이션이 되고 대상 클러스터에서 실행 중인 애플리케이션에 적용된 모든 변경 사항은 삭제됩니다.

이 프로세스는 복제를 재설정하기 전에 대상 클러스터에서 앱을 중지합니다.

장애 조치 중에 대상 애플리케이션에 기록된 모든 데이터는 손실됩니다.

단계

1. 선택 사항: 소스 클러스터에서 소스 애플리케이션의 스냅샷을 만듭니다. 이렇게 하면 소스 클러스터의 최신 변경 사항이 캡처됩니다.
2. 대상 클러스터에서 AppMirrorRelationship CR 파일을 편집합니다(예: `trident-protect-relationship.yaml`) 및 spec.desiredState의 값을 다음으로 변경합니다. `Established` .
3. CR 파일을 저장합니다.
4. CR을 적용하세요:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 장애 조치된 애플리케이션을 보호하기 위해 대상 클러스터에 보호 일정을 만든 경우 해당 일정을 제거합니다. 남아 있는 일정으로 인해 볼륨 스냅샷 오류가 발생합니다.

장애가 발생한 복제 관계의 역방향 재동기화

장애 조치된 복제 관계를 역방향 재동기화하면 대상 애플리케이션이 소스 애플리케이션이 되고, 소스가 대상이 됩니다. 장애 조치 중에 대상 애플리케이션에 적용된 변경 사항은 유지됩니다.

단계

1. 원래 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다. 이로 인해 목적지가 소스가 됩니다. 새 대상 클러스터에 보호 일정이 남아 있으면 제거합니다.
2. 원래 관계를 설정하는 데 사용했던 CR 파일을 반대 클러스터에 적용하여 복제 관계를 설정합니다.

3. 새로운 대상(원래 소스 클러스터)이 두 AppVault CR로 구성되었는지 확인하세요.
4. 반대쪽 클러스터에 복제 관계를 설정하고 역방향에 대한 값을 구성합니다.

역방향 애플리케이션 복제 방향

복제 방향을 반대로 하면 Trident Protect는 원래 소스 스토리지 백엔드로 복제를 계속 진행하면서 애플리케이션을 대상 스토리지 백엔드로 이동합니다. Trident Protect는 소스 애플리케이션을 중지하고 대상 앱으로 장애 조치하기 전에 대상에 데이터를 복제합니다.

이 상황에서는 소스와 목적지가 바뀌게 됩니다.

단계

1. 소스 클러스터에서 종료 스냅샷을 만듭니다.

CR을 사용하여 종료 스냅샷을 만듭니다.

a. 소스 애플리케이션에 대한 보호 정책 일정을 비활성화합니다.

b. ShutdownSnapshot CR 파일을 만듭니다.

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-shutdownsnapshot.yaml`).

ii. 다음 속성을 구성합니다.

- **metadata.name**: (필수) 사용자 정의 리소스의 이름입니다.
- **spec.AppVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.ApplicationRef**: (필수) 이 값은 소스 애플리케이션 CR 파일의 metadata.name 필드와 일치해야 합니다.

YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. 다음을 채운 후 `trident-protect-shutdownsnapshot.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

CLI를 사용하여 종료 스냅샷을 만듭니다.

a. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 종료 스냅샷을 만듭니다. 예를 들어:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. 소스 클러스터에서 종료 스냅샷이 완료된 후 종료 스냅샷의 상태를 가져옵니다.

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. 소스 클러스터에서 다음 명령을 사용하여 *shutdownsnapshot.status.appArchivePath*의 값을 찾고 파일 경로의 마지막 부분(기본 이름이라고도 함. 마지막 슬래시 뒤의 모든 내용이 됩니다)을 기록합니다.

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. 다음 변경 사항을 적용하여 새 대상 클러스터에서 새 소스 클러스터로 장애 조치를 수행합니다.

> 장애 조치 절차의 2단계에서 다음을 포함합니다. `spec.promotedSnapshot` AppMirrorRelationship CR 파일의 필드를 설정하고 해당 값을 위 3단계에서 기록한 기본 이름으로 설정합니다.

5. 역방향 재동기화 단계를 수행하십시오.장애가 발생한 복제 관계의 역방향 재동기화 .
6. 새 소스 클러스터에서 보호 일정을 활성화합니다.

결과

다음 동작은 역방향 복제로 인해 발생합니다.

- 원본 소스 앱의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 원래 소스 앱의 Pod는 앱의 Kubernetes 리소스를 삭제하여 정상적으로 중지됩니다(PVC와 PV는 그대로 유지).
- 포드가 종료된 후 앱 볼륨의 스냅샷이 촬영되어 복제됩니다.
- SnapMirror 관계가 끊어져 대상 볼륨을 읽기/쓰기할 수 있게 되었습니다.
- 앱의 Kubernetes 리소스는 원래 소스 앱이 종료된 후 복제된 볼륨 데이터를 사용하여 종료 전 스냅샷에서 복원됩니다.
- 복제는 역방향으로 재설정됩니다.

원래 소스 클러스터로 애플리케이션을 장애 복구합니다.

Trident Protect를 사용하면 다음과 같은 작업 순서를 통해 장애 조치 작업 후에 "장애 복구"를 달성할 수 있습니다. 원래 복제 방향을 복원하기 위한 이 워크플로에서 Trident Protect는 복제 방향을 반전하기 전에 모든 애플리케이션 변경 사항을 원래 소스 애플리케이션으로 복제(재동기화)합니다.

이 프로세스는 대상에 대한 장애 조치를 완료한 관계에서 시작되며 다음 단계를 포함합니다.

- 실패한 상태로 시작합니다.
- 복제 관계를 역방향으로 재동기화합니다.

장애 조치 절차 중에 대상 클러스터에 기록된 데이터가 삭제되므로 일반적인 재동기화 작업을 수행하지 마세요.

- 복제 방향을 반대로 합니다.

단계

1. 수행하다장애가 발생한 복제 관계의 역방향 재동기화 단계.
2. 수행하다역방향 애플리케이션 복제 방향 단계.

복제 관계 삭제

언제든지 복제 관계를 삭제할 수 있습니다. 애플리케이션 복제 관계를 삭제하면 서로 관계가 없는 두 개의 별도 애플리케이션이 생성됩니다.

단계

1. 현재 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다.

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Trident Protect를 사용하여 애플리케이션 마이그레이션

백업 데이터를 복원하여 클러스터 간이나 다른 스토리지 클래스로 애플리케이션을 마이그레이션할 수 있습니다.

애플리케이션을 마이그레이션하면 해당 애플리케이션에 구성된 모든 실행 후크가 앱과 함께 마이그레이션됩니다. 복원 후 실행 후크가 있는 경우 복원 작업의 일부로 자동으로 실행됩니다.

백업 및 복원 작업

다음 시나리오에 대한 백업 및 복원 작업을 수행하려면 특정 백업 및 복원 작업을 자동화할 수 있습니다.

동일한 클러스터에 복제

동일한 클러스터에 애플리케이션을 복제하려면 스냅샷이나 백업을 생성하고 동일한 클러스터에 데이터를 복원합니다.

단계

1. 다음 중 하나를 수행하세요.
   a. "스냅샷 만들기".
   b. "백업을 만듭니다".
2. 동일한 클러스터에서 스냅샷이나 백업을 생성했는지에 따라 다음 중 하나를 수행합니다.
   a. "스냅샷에서 데이터 복원".
   b. "백업에서 데이터를 복원하세요".

다른 클러스터에 애플리케이션을 복제하려면(클러스터 간 복제 수행) 소스 클러스터에 백업을 만든 다음, 해당 백업을 다른 클러스터로 복원합니다. 대상 클러스터에 Trident Protect가 설치되어 있는지 확인하세요.

다음을 사용하여 다른 클러스터 간에 애플리케이션을 복제할 수 있습니다."SnapMirror 복제" .

단계

1. "백업을 만듭니다".
2. 백업이 포함된 개체 스토리지 버킷에 대한 AppVault CR이 대상 클러스터에 구성되었는지 확인하세요.
3. 대상 클러스터에서"백업에서 데이터를 복원합니다" .

한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션 마이그레이션

대상 스토리지 클래스에 백업을 복원하여 애플리케이션을 한 스토리지 클래스에서 다른 스토리지 클래스로 마이그레이션할 수 있습니다.

예를 들어 (복원 CR의 비밀 제외):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

CR을 사용하여 스냅샷을 복원합니다.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-snapshot-restore-cr.yaml` .
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (필수) 스냅샷 내용이 저장된 AppVault의 이름입니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 대상 네임스페이스로 매핑합니다. 바꾸다 `my-source-namespace` 그리고 `my-destination-namespace` 주변 환경의 정보를 활용하여

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. 선택적으로, 복원할 애플리케이션의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필요) 사용 `include or exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 다음 resourceMatchers 매개변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 OR 연산으로 일치하고, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스 그룹입니다.
       - **resourceMatchers[].kind**: (선택 사항) 필터링할 리소스의 종류입니다.
       - **resourceMatchers[].version**: (선택 사항) 필터링할 리소스의 버전입니다.

- **resourceMatchers[].names**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
- **resourceMatchers[].namespaces**: (선택 사항) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
- **resourceMatchers[].labelSelectors**: (선택 사항) 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "쿠버네티스 문서" . 예를 들어: `"trident.netapp.io/os=linux"` .

  예를 들어:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. 다음을 채운 후 `trident-protect-snapshot-restore-cr.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI를 사용하여 스냅샷 복원

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔서 스냅샷을 다른 네임스페이스로 복원합니다.
   - 그만큼 `snapshot` 인수는 형식에 네임스페이스와 스냅샷 이름을 사용합니다. `<namespace>/<name>` .
   - 그만큼 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 형식에 맞는 올바른 대상 네임스페이스에 매핑합니다. `source1:dest1,source2:dest2` .

     예를 들어:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Trident Protect 실행 후크 관리

실행 후크는 관리되는 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어, 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 이를 통해 애플리케이션과 일관된 스냅샷이 보장됩니다.

실행 후크의 유형

Trident Protect는 실행 가능 시기에 따라 다음 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 스냅샷 이후
- 사전 백업
- 백업 후
- 복원 후
- 장애 조치 후

실행 순서

데이터 보호 작업이 실행되면 실행 후크 이벤트는 다음 순서로 발생합니다.

1. 적용 가능한 사용자 정의 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 정의 사전 작업 후크를 만들고 실행할 수 있지만, 작업 전에 이러한 후크를 실행하는 순서는 보장되지 않으며 구성할 수도 없습니다.
2. 해당되는 경우 파일 시스템이 정지됩니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".
3. 데이터 보호 작업이 수행됩니다.
4. 해당되는 경우 동결된 파일 시스템이 동결 해제됩니다.
5. 적용 가능한 모든 사용자 정의 사후 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 정의 사후 작업 후크를 만들고 실행할 수 있지만, 작업 후 이러한 후크의 실행 순서는 보장되지 않으며 구성할 수도 없습니다.

동일한 유형의 실행 후크를 여러 개 생성하는 경우(예: 사전 스냅샷), 해당 후크의 실행 순서는 보장되지 않습니다. 하지만 서로 다른 유형의 후크 실행 순서는 보장됩니다. 예를 들어, 다양한 유형의 후크를 모두 포함하는 구성을 실행하는 순서는 다음과 같습니다.

1. 스냅샷 전 후크 실행됨
2. 스냅샷 후 후크 실행됨
3. 사전 백업 후크 실행됨

4. 백업 후 후크 실행됨

이전 주문 예시는 기존 스냅샷을 사용하지 않는 백업을 실행할 때만 적용됩니다.

프로덕션 환경에서 실행 후크 스크립트를 활성화하기 전에 항상 테스트해야 합니다. 'kubectl exec' 명령을 사용하면 편리하게 스크립트를 테스트할 수 있습니다. 프로덕션 환경에서 실행 후크를 활성화한 후 결과 스냅샷과 백업을 테스트하여 일관성이 있는지 확인하세요. 앱을 임시 네임스페이스에 복제하고 스냅샷이나 백업을 복원한 다음 앱을 테스트하면 됩니다.

스냅샷 이전 실행 후크가 Kubernetes 리소스를 추가, 변경 또는 제거하는 경우 해당 변경 사항은 스냅샷이나 백업 및 이후의 모든 복원 작업에 포함됩니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱의 실행 후크를 계획할 때 다음 사항을 고려하세요.

- 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 여러 개의 실행 후크가 동일한 스크립트를 참조할 수 있습니다.
- Trident Protect에서는 실행 후크가 사용하는 스크립트가 실행 가능한 셸 스크립트 형식으로 작성되어야 합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Trident Protect는 실행 후크 설정과 일치 기준을 사용하여 스냅샷, 백업 또는 복원 작업에 적용할 수 있는 후크를 결정합니다.

실행 후크는 종종 실행 중인 애플리케이션의 기능을 감소시키거나 완전히 비활성화하기 때문에 사용자 정의 실행 후크가 실행되는 데 걸리는 시간을 최소화하려고 항상 노력해야 합니다. 연관된 실행 후크로 백업 또는 스냅샷 작업을 시작한 후 취소하더라도 백업 또는 스냅샷 작업이 이미 시작된 경우 후크는 계속 실행될 수 있습니다. 즉, 백업 후 실행 후크에 사용된 로직은 백업이 완료되었다고 가정할 수 없습니다.

실행 후크 필터

애플리케이션에 대한 실행 후크를 추가하거나 편집할 때 실행 후크에 필터를 추가하여 후크가 일치할 컨테이너를 관리할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하지만 각 이미지를 다른 목적으로 사용할 수 있는 애플리케이션(예: Elasticsearch)에 유용합니다. 필터를 사용하면 실행 후크가 일부 컨테이너에서만 실행되는 시나리오를 만들 수 있지만 반드시 모든 컨테이너에서 실행되는 것은 아닙니다. 단일 실행 후크에 대해 여러 필터를 만드는 경우 논리적 AND 연산자를 사용하여 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 가질 수 있습니다.

실행 후크에 추가하는 각 필터는 정규 표현식을 사용하여 클러스터의 컨테이너와 일치시킵니다. 후크가 컨테이너와 일치하면 후크는 해당 컨테이너에서 연관된 스크립트를 실행합니다. 필터에 대한 정규 표현식은 RE2(Regular Expression 2) 구문을 사용하는데, 이 구문은 일치 항목 목록에서 컨테이너를 제외하는 필터를 만드는 것을 지원하지 않습니다. Trident Protect가 실행 후크 필터의 정규 표현식에 대해 지원하는 구문에 대한 정보는 다음을 참조하세요. "정규 표현식 2(RE2) 구문 지원" .

복원 또는 복제 작업 후에 실행되는 실행 후크에 네임스페이스 필터를 추가하고 복원 또는 복제 소스와 대상이 다른 네임스페이스에 있는 경우, 네임스페이스 필터는 대상 네임스페이스에만 적용됩니다.

실행 후크 예제

방문하세요 "NetApp Verda GitHub 프로젝트" Apache Cassandra 및 Elasticsearch와 같은 인기 있는 앱에 대한 실제 실행 후크를 다운로드합니다. 또한 사용자 정의 실행 후크를 구성하는 데 필요한 예제를 보고 아이디어를 얻을 수 있습니다.

실행 후크 만들기

예 **1.** 을 사용하여 앱에 대한 사용자 정의 실행 후크를 만들 수 있습니다**.** 실행 후크를 생성하려면 소유자**,** 관리자 또는 멤버 권한이 필요합니다**.**

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-hook.yaml` .
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 실행 후크를 실행할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.stage**: (필수) 실행 후크가 동작 중 어느 단계에서 실행되어야 하는지를 나타내는 문자열입니다. 가능한 값:
     - 사전
     - 우편
   - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 지원
     - 복원하다
     - 장애 조치
   - **spec.enabled**: (선택 사항) 이 실행 후크가 활성화되어 있는지 비활성화되어 있는지를 나타냅니다. 지정하지 않으면 기본값은 true입니다.
   - **spec.hookSource**: (필수) base64로 인코딩된 후크 스크립트가 포함된 문자열입니다.
   - **spec.timeout**: (선택 사항) 실행 후크가 실행될 수 있는 시간을 분 단위로 정의하는 숫자입니다. 최소값은 1분이며, 지정하지 않으면 기본값은 25분입니다.
   - **spec.arguments**: (선택 사항) 실행 후크에 지정할 수 있는 인수의 YAML 목록입니다.
   - **spec.matchingCriteria**: (선택 사항) 조건 키 값 쌍의 선택적 목록으로, 각 쌍은 실행 후크 필터를 구성합니다. 실행 후크당 최대 10개의 필터를 추가할 수 있습니다.
   - **spec.matchingCriteria.type**: (선택 사항) 실행 후크 필터 유형을 식별하는 문자열입니다. 가능한 값:
     - 컨테이너이미지
     - 컨테이너 이름
     - 포드이름
     - 포드레이블
     - 네임스페이스 이름
   - **spec.matchingCriteria.value**: (선택 사항) 실행 후크 필터 값을 식별하는 문자열 또는 정규 표현식입니다.

     YAML 예시:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. CR 파일에 올바른 값을 채운 후 CR을 적용합니다.

```
kubectl apply -f trident-protect-hook.yaml
```

CLI를 사용하세요

단계

1. 괄호 안의 값을 환경의 정보로 바꿔서 실행 후크를 만듭니다. 예를 들어:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

수동으로 실행 후크 실행

테스트 목적으로 실행 후크를 수동으로 실행할 수 있으며, 실패 후 후크를 수동으로 다시 실행해야 하는 경우에도 사용할 수 있습니다. 실행 후크를 수동으로 실행하려면 소유자, 관리자 또는 멤버 권한이 필요합니다.

실행 후크를 수동으로 실행하는 것은 두 가지 기본 단계로 구성됩니다.

1. 리소스를 수집하고 백업을 생성하여 후크가 실행될 위치를 결정하는 리소스 백업을 생성합니다.
2. 백업에 대해 실행 후크를 실행합니다.

1단계: 리소스 백업 만들기

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-resource-backup.yaml` .
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 리소스 백업을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 내용이 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 백업 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     YAML 예시:

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: ResourceBackup
     metadata:
       name: example-resource-backup
     spec:
       applicationRef: my-app-name
       appVaultRef: my-appvault-name
       appArchivePath: example-resource-backup
     ```

3. CR 파일에 올바른 값을 채운 후 CR을 적용합니다.

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

CLI를 사용하세요

단계

1. 괄호 안의 값을 사용자 환경의 정보로 바꿔 백업을 만듭니다. 예를 들어:

   ```
   tridentctl protect create resourcebackup <my_backup_name> --app
   <my_app_name> --appvault <my_appvault_name> -n
   <my_app_namespace> --app-archive-path <app_archive_path>
   ```

2. 백업 상태를 확인합니다. 작업이 완료될 때까지 이 예제 명령을 반복해서 사용할 수 있습니다.

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. 백업이 성공했는지 확인하세요.

```
kubectl describe resourcebackup <my_backup_name>
```

2단계: 실행 후크 실행

CR을 사용하세요

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다. `trident-protect-hook-run.yaml` .
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 이 값이 1단계에서 만든 ResourceBackup CR의 애플리케이션 이름과 일치하는지 확인하세요.
   - **spec.appVaultRef**: (필수) 이 값이 1단계에서 생성한 ResourceBackup CR의 appVaultRef와 일치하는지 확인하세요.
   - **spec.appArchivePath**: 이 값이 1단계에서 만든 ResourceBackup CR의 appArchivePath와 일치하는지 확인합니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 지원
     - 복원하다
     - 장애 조치
   - **spec.stage**: (필수) 실행 후크가 동작 중 어느 단계에서 실행되어야 하는지를 나타내는 문자열입니다. 이 후크 실행은 다른 단계에서 후크를 실행하지 않습니다. 가능한 값:
     - 사전
     - 우편

     YAML 예시:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. CR 파일에 올바른 값을 채운 후 CR을 적용합니다.

```
kubectl apply -f trident-protect-hook-run.yaml
```

CLI를 사용하세요

단계

1. 수동 실행 후크 실행 요청을 만듭니다.

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. 실행 후크 실행 상태를 확인합니다. 작업이 완료될 때까지 이 명령을 반복해서 실행할 수 있습니다.

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. 최종 세부 정보와 상태를 확인하려면 exechooksrun 객체를 설명합니다.

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Trident Protect 제거

평가판에서 정식 버전으로 업그레이드하는 경우 Trident Protect 구성 요소를 제거해야 할 수도 있습니다.

Trident Protect를 제거하려면 다음 단계를 수행하세요.

단계

1. Trident Protect CR 파일을 제거합니다.

   이 단계는 25.06 이상 버전에서는 필요하지 않습니다.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Trident Protect 제거:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Trident Protect 네임스페이스를 제거합니다.

   ```
   kubectl delete ns trident-protect
   ```

Trident 및 Trident 프로텍트 블로그

훌륭한 NetApp Trident 및 Trident Protect 블로그는 다음에서 찾을 수 있습니다.

Trident 블로그

- 2025년 5월 9일:"Amazon EKS 애드온을 사용한 FSx for ONTAP 에 대한 자동 Trident 백엔드 구성"
- 2025년 8월 19일:"데이터 일관성 향상: Trident 사용한 OpenShift 가상화의 볼륨 그룹 스냅샷"
- 2025년 4월 15일:"Google Cloud NetApp Volumes for SMB Protocol을 갖춘 NetApp Trident"
- 2025년 4월 14일:"Kubernetes에서 영구 저장소를 위해 Trident 25.02와 함께 파이버 채널 프로토콜 활용"
- 2025년 4월 14일:"Kubernetes 블록 스토리지를 위한 NetApp ASA r2 시스템의 잠재력 활용"
- 2025년 3월 31일:"새로운 인증 운영자를 통해 Red Hat OpenShift에서 Trident 설치 간소화"
- 2025년 3월 27일:"Google Cloud NetApp Volumes 사용하여 SMB용 Trident 프로비저닝"
- 2025년 3월 5일:"원활한 iSCSI 스토리지 통합 잠금 해제: AWS용 ROSA 클러스터에서 FSxN 가이드"
- 2025년 2월 27일:"Trident, GKE 및 Google Cloud NetApp Volumes 사용하여 클라우드 ID 배포"
- 2024년 12월 12일:"Trident 에서 파이버 채널 지원 소개"
- 2024년 11월 26일:"Trident 25.01: 새로운 기능과 향상된 기능으로 Kubernetes 스토리지 환경 향상"
- 2024년 11월 11일:"Google Cloud NetApp Volumes 탑재한 NetApp Trident"
- 2024년 10월 29일:"Trident 사용하여 AWS(ROSA)에서 Red Hat OpenShift 서비스를 사용하는 Amazon FSx for NetApp ONTAP"
- 2024년 10월 29일:"ROSA 및 Amazon FSx for NetApp ONTAP 에서 OpenShift Virtualization을 사용한 VM 라이브 마이그레이션"
- 2024년 7월 8일:"Amazon EKS에서 최신 컨테이너화된 앱에 대한 ONTAP 스토리지를 사용하기 위해 NVMe/TCP 사용"
- 2024년 7월 1일:"Google Cloud NetApp Volumes Flex 및 Astra Trident 활용한 원활한 Kubernetes 스토리지"
- 2024년 6월 11일:"OpenShift의 통합 이미지 레지스트리를 위한 백엔드 스토리지로서의 ONTAP"

Trident 프로텍트 블로그

- 2025년 5월 16일:"Trident Protect 복원 후 후크를 사용하여 재해 복구를 위한 레지스트리 장애 조치 자동화"
- 2025년 5월 16일:"NetApp Trident Protect를 통한 OpenShift 가상화 재해 복구"
- 2025년 5월 13일:"Trident Protect 백업 및 복원을 통한 스토리지 클래스 마이그레이션"
- 2025년 5월 9일:"Trident Protect 복원 후 후크를 사용하여 Kubernetes 애플리케이션 크기 조정"
- 2025년 4월 3일:"Trident Protect Power Up: 보호 및 재해 복구를 위한 Kubernetes 복제"
- 2025년 3월 13일:"OpenShift Virtualization VM에 대한 충돌 일관성 백업 및 복원 작업"
- 2025년 3월 11일:"NetApp Trident 사용하여 GitOps 패턴을 애플리케이션 데이터 보호로 확장"
- 2025년 3월 3일:"Trident 25.02: 흥미로운 새 기능으로 Red Hat OpenShift 경험 향상"

- 2025년 1월 15일:"Trident Protect 역할 기반 액세스 제어 소개"
- 2024년 11월 11일: "Trident Protect를 위한 강력한 CLI인 tridentctl protect를 소개합니다."
- 2024년 11월 11일:"Kubernetes 기반 데이터 관리: Trident Protect로 시작하는 새로운 시대"

지식과 지원

자주 묻는 질문

Trident 설치, 구성, 업그레이드 및 문제 해결에 대한 자주 묻는 질문에 대한 답변을 찾아보세요.

일반적인 질문

Trident 얼마나 자주 출시되나요**?**

24.02 릴리스부터 Trident 4개월마다(2월, 6월, 10월) 출시됩니다.

Trident Kubernetes의 특정 버전에서 출시된 모든 기능을 지원합니까**?**

Trident 일반적으로 Kubernetes의 알파 기능을 지원하지 않습니다. Trident Kubernetes 베타 릴리스 이후에 나오는 두 가지 Trident 릴리스에서 베타 기능을 지원할 수 있습니다.

Trident 의 작동을 위해 다른 **NetApp** 제품에 대한 종속성이 있습니까**?**

Trident 다른 NetApp 소프트웨어 제품에 대한 종속성이 없으며 독립 실행형 애플리케이션으로 작동합니다. 하지만 NetApp 백엔드 스토리지 장치가 있어야 합니다.

Trident 구성에 대한 자세한 내용을 어떻게 얻을 수 있나요**?**

사용하다 `tridentctl get` Trident 구성에 대한 자세한 정보를 얻으려면 명령을 사용하세요.

Trident 가 스토리지를 프로비저닝하는 방식에 대한 메트릭을 얻을 수 있나요**?**

네. Trident 작업에 대한 정보(관리되는 백엔드 수, 프로비저닝된 볼륨 수, 사용된 바이트 수 등)를 수집하는 데 사용할 수 있는 Prometheus 엔드포인트입니다. 또한 사용할 수 있습니다"Cloud Insights" 모니터링 및 분석을 위해.

Trident CSI Provisioner로 사용하면 사용자 경험이 달라지나요**?**

아니요. 사용자 경험과 기능 면에서는 아무런 변화가 없습니다. 사용된 프로비저너 이름은 다음과 같습니다. `csi.trident.netapp.io` . 현재 및 향후 릴리스에서 제공되는 모든 새로운 기능을 사용하려면 이 Trident 설치 방법을 사용하는 것이 좋습니다.

Kubernetes 클러스터에 Trident 설치 및 사용

Trident 개인 레지스트리에서 오프라인 설치를 지원합니까**?**

네, Trident 오프라인으로 설치할 수 있습니다. 참조하다"Trident 설치에 대해 알아보세요" .

원격으로 **Trident** 설치할 수 있나요**?**

네. Trident 18.10 이상은 모든 컴퓨터에서 원격 설치 기능을 지원합니다. `kubectl` 클러스터에 접근합니다. 후에 `kubectl` 액세스가 검증되었습니다(예: 시작 `kubectl get nodes` 원격 컴퓨터에서 명령을 입력하여 확인하고, 설치 지침을 따르세요.

Trident 로 고가용성을 구성할 수 있나요?

Trident 인스턴스 하나가 있는 Kubernetes 배포(ReplicaSet)로 설치되므로 HA가 내장되어 있습니다. 배포에서 복제본 수를 늘리면 안 됩니다. Trident 설치된 노드가 손실되거나 Pod에 다른 방법으로 접근할 수 없는 경우, Kubernetes는 클러스터 내의 정상적인 노드에 Pod를 자동으로 다시 배포합니다. Trident 제어 평면에만 적용되므로 Trident 재배치되더라도 현재 장착된 포드는 영향을 받지 않습니다.

Trident kube-system 네임스페이스에 액세스해야 합니까?

Trident 애플리케이션이 새로운 PVC를 요청하는 시점을 확인하기 위해 Kubernetes API 서버에서 데이터를 읽어오므로 kube-system에 액세스해야 합니다.

Trident 에서 사용하는 역할과 권한은 무엇입니까?

Trident 설치 프로그램은 Kubernetes 클러스터의 PersistentVolume, PersistentVolumeClaim, StorageClass 및 Kubernetes 클러스터의 Secret 리소스에 대한 특정 액세스 권한이 있는 Kubernetes ClusterRole을 생성합니다. 참조하다"tridentctl 설치 사용자 정의" .

Trident 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성할 수 있나요?

필요한 경우 Trident 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성하고 수정할 수 있습니다. 참조하다"tridentctl 설치 사용자 정의" .

두 개의 별도 Kubernetes 클러스터에 대해 두 개의 별도 Trident 인스턴스에 대해 동일한 ONTAP 백엔드 SVM을 공유할 수 있습니까?

권장하지는 않지만 두 개의 Trident 인스턴스에 동일한 백엔드 SVM을 사용할 수 있습니다. 설치 중 각 인스턴스에 대해 고유한 볼륨 이름을 지정하거나 고유한 볼륨 이름을 지정하십시오. `StoragePrefix` 매개변수 `setup/backend.json` 파일. 이는 동일한 FlexVol volume 이 두 인스턴스 모두에 사용되지 않도록 하기 위한 것입니다.

ContainerLinux(이전 CoreOS)에 Trident 설치할 수 있나요?

Trident 단순히 Kubernetes 포드일 뿐이며 Kubernetes가 실행되는 곳이라면 어디에나 설치할 수 있습니다.

Trident NetApp Cloud Volumes ONTAP 과 함께 사용할 수 있나요?

네, Trident AWS, Google Cloud, Azure에서 지원됩니다.

Trident Cloud Volumes Services와 호환되나요?

네, Trident Azure의 Azure NetApp Files 서비스와 GCP의 Cloud Volumes Service 지원합니다.

문제 해결 및 지원

NetApp Trident 지원합니까?

Trident 는 오픈 소스이고 무료로 제공되지만 NetApp 백엔드가 지원되는 경우 NetApp 에서도 Trident를 완벽하게 지원합니다.

지원 사례를 제기하려면 어떻게 해야 하나요?

지원 사례를 제기하려면 다음 중 하나를 수행하세요.

1. 지원 계정 관리자에게 문의하여 티켓 제출에 대한 도움을 받으세요.
2. 지원 사례를 문의하세요. "NetApp 지원" .

지원 로그 번들을 어떻게 생성하나요?

다음을 실행하여 지원 번들을 생성할 수 있습니다. `tridentctl logs -a` . 번들에 캡처된 로그 외에도 kubelet 로그를 캡처하여 Kubernetes 측의 마운트 문제를 진단합니다. kubelet 로그를 얻는 방법은 Kubernetes가 설치된 방법에 따라 다릅니다.

새로운 기능에 대한 요청을 해야 할 경우 어떻게 해야 하나요?

문제를 생성하세요 "Trident Github" 그리고 해당 이슈의 제목과 설명에 *RFE*를 언급하세요.

결함은 어디에 제기해야 하나요?

문제를 생성하세요 "Trident Github" . 해당 문제와 관련된 모든 필수 정보와 로그를 포함시키세요.

Trident 에 대해 명확히 알고 싶은 질문이 있으면 어떻게 해야 하나요? 커뮤니티나 포럼이 있나요?

질문, 문제 또는 요청 사항이 있는 경우 Trident 통해 문의해 주세요."디스코드 채널" 또는 GitHub.

저장 시스템의 비밀번호가 변경되어 Trident 더 이상 작동하지 않습니다. 어떻게 복구할 수 있나요?

백엔드의 비밀번호를 업데이트하세요. `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident` . 바꾸다 `myBackend` 백엔드 이름을 사용한 예에서 `/path/to_new_backend.json` 올바른 길로 가는 길 `backend.json` 파일.

Trident 내 Kubernetes 노드를 찾을 수 없습니다. 이 문제를 어떻게 해결하나요?

Trident Kubernetes 노드를 찾을 수 없는 데에는 두 가지 시나리오가 있을 수 있습니다. Kubernetes 내부의 네트워킹 문제나 DNS 문제 때문일 수 있습니다. 각 Kubernetes 노드에서 실행되는 Trident 노드 데몬셋은 Trident 컨트롤러와 통신하여 노드를 Trident 에 등록할 수 있어야 합니다. Trident 설치한 후 네트워킹이 변경된 경우, 클러스터에 추가된 새로운 Kubernetes 노드에서만 이 문제가 발생합니다.

Trident 포드가 파괴되면 데이터가 손실되나요?

Trident 포드가 파괴되어도 데이터는 손실되지 않습니다. Trident 메타데이터는 CRD 객체에 저장됩니다. Trident 에서 프로비저닝한 모든 PV는 정상적으로 작동합니다.

Trident 업그레이드

이전 버전에서 새 버전으로 바로 업그레이드할 수 있나요(몇 가지 버전을 건너뛸 수 있나요)?

NetApp Trident 현재 주요 릴리스에서 바로 다음 주요 릴리스로 업그레이드하는 것을 지원합니다. 버전 18.xx에서 19.xx로, 19.xx에서 20.xx로 업그레이드할 수 있습니다. 실제 운영에 적용하기 전에 실험실에서 업그레이드를 테스트해야 합니다.

Trident 이전 릴리스로 다운그레이드할 수 있나요**?**

업그레이드 후 발견된 버그, 종속성 문제 또는 실패하거나 완료되지 않은 업그레이드에 대한 수정이 필요한 경우 다음을 수행해야 합니다."Trident 제거" 그리고 해당 버전에 대한 구체적인 지침을 사용하여 이전 버전을 다시 설치하세요. 이는 이전 버전으로 다운그레이드하는 데 권장되는 유일한 방법입니다.

백엔드 및 볼륨 관리

ONTAP 백엔드 정의 파일에서 **Management**와 **DataLIF**를 모두 정의해야 합니까**?**

관리 LIF는 필수입니다. DataLIF는 다음과 같이 다양합니다.

- ONTAP SAN: iSCSI에 대해서는 지정하지 마세요. Trident 사용"ONTAP 선택적 LUN 맵" 다중 경로 세션을 설정하는 데 필요한 iSCI LIF를 발견합니다. 경고가 생성됩니다. `dataLIF` 명확하게 정의되어 있습니다. 참조하다 "ONTAP SAN 구성 옵션 및 예" 자세한 내용은.
- ONTAP NAS: NetApp 다음을 지정하는 것을 권장합니다. `dataLIF` . 제공되지 않으면 Trident SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하면 라운드 로빈 DNS를 만들어 여러 dataLIF에 걸쳐 부하를 분산할 수 있습니다. 참조하다"ONTAP NAS 구성 옵션 및 예" 자세한 내용은

Trident ONTAP 백엔드에 대해 **CHAP**를 구성할 수 있나요**?**

네. Trident ONTAP 백엔드에 대해 양방향 CHAP를 지원합니다. 여기에는 설정이 필요합니다 `useCHAP=true` 백엔드 구성에서.

Trident 사용하여 수출 정책을 어떻게 관리합니까**?**

Trident 20.04 버전부터 동적으로 내보내기 정책을 만들고 관리할 수 있습니다. 이를 통해 스토리지 관리자는 백엔드 구성에서 하나 이상의 CIDR 블록을 제공하고 Trident 해당 범위에 속하는 노드 IP를 생성한 내보내기 정책에 추가할 수 있습니다. 이런 방식으로 Trident 주어진 CIDR 내의 IP를 가진 노드에 대한 규칙의 추가 및 삭제를 자동으로 관리합니다.

IPv6 주소를 **Management** 및 **DataLIF**에 사용할 수 있나요**?**

Trident 다음에 대한 IPv6 주소 정의를 지원합니다.

- `managementLIF`그리고 `dataLIF` ONTAP NAS 백엔드용.
- `managementLIF`ONTAP SAN 백엔드용. 지정할 수 없습니다 `dataLIF` ONTAP SAN 백엔드에서.

Trident 플래그를 사용하여 설치해야 합니다. `--use-ipv6` (을 위한 `tridentctl` 설치), `IPv6` (Trident 운영자의 경우) 또는 `tridentTPv6` (Helm 설치의 경우) IPv6에서 작동하도록 해야 합니다.

백엔드에서 **Management LIF**를 업데이트할 수 있나요**?**

예, 다음을 사용하여 백엔드 관리 LIF를 업데이트할 수 있습니다. `tridentctl update backend` 명령.

백엔드에서 **DataLIF**를 업데이트하는 것이 가능합니까**?**

DataLIF를 업데이트할 수 있습니다. `ontap-nas` 그리고 `ontap-nas-economy` 오직.

Kubernetes용 **Trident** 에서 여러 개의 백엔드를 만들 수 있나요**?**

Trident 동일한 드라이버나 다른 드라이버를 사용하여 여러 백엔드를 동시에 지원할 수 있습니다.

Trident 백엔드 자격 증명을 어떻게 저장합니까**?**

Trident 백엔드 자격 증명을 Kubernetes Secrets로 저장합니다.

Trident 어떻게 특정 백엔드를 선택하나요**?**

백엔드 속성을 사용하여 클래스에 적합한 풀을 자동으로 선택할 수 없는 경우 `storagePools` 그리고 `additionalStoragePools` 매개변수는 특정 풀 세트를 선택하는 데 사용됩니다.

Trident 특정 백엔드에서 프로비저닝을 하지 않도록 하려면 어떻게 해야 하나요**?**

그만큼 `excludeStoragePools` 매개변수는 Trident 프로비저닝에 사용하는 풀 세트를 필터링하는 데 사용되며 일치하는 모든 풀을 제거합니다.

같은 종류의 백엔드가 여러 개 있는 경우**, Trident** 어떤 백엔드를 사용할지 어떻게 선택합니까**?**

동일한 유형의 구성된 백엔드가 여러 개 있는 경우 Trident 현재 매개변수를 기반으로 적절한 백엔드를 선택합니다. `StorageClass` 그리고 `PersistentVolumeClaim` . 예를 들어, 여러 개의 ontap-nas 드라이버 백엔드가 있는 경우 Trident 매개변수를 일치시키려고 시도합니다. `StorageClass` 그리고 `PersistentVolumeClaim` 결합되어 요구 사항을 충족할 수 있는 백엔드와 일치합니다. `StorageClass` 그리고 `PersistentVolumeClaim` . 요청과 일치하는 백엔드가 여러 개 있는 경우 Trident 그 중 하나를 무작위로 선택합니다.

Trident Element/ SolidFire 와 양방향 **CHAP**를 지원합니까**?**

네.

Trident ONTAP 볼륨에 **Qtrees**를 어떻게 배포합니까**?** 단일 볼륨에 몇 개의 **Qtree**를 배포할 수 있나요**?**

그만큼 `ontap-nas-economy` 드라이버는 동일한 FlexVol volume 에 최대 200개의 Qtree를 생성하고(50~300개로 구성 가능), 클러스터 노드당 100,000개의 Qtree, 클러스터당 2.4M개의 Qtree를 생성합니다. 새로운 것을 입력할 때 `PersistentVolumeClaim` 경제 드라이버가 서비스하는 경우 드라이버는 새 Qtree를 서비스할 수 있는 FlexVol volume 이미 있는지 확인합니다. Qtree에 서비스를 제공할 수 있는 FlexVol volume 없으면 새로운 FlexVol volume 생성됩니다.

ONTAP NAS에 프로비저닝된 볼륨에 대한 **Unix** 권한을 어떻게 설정할 수 있나요**?**

백엔드 정의 파일에서 매개변수를 설정하여 Trident 에서 프로비저닝한 볼륨에 대한 Unix 권한을 설정할 수 있습니다.

볼륨을 프로비저닝하는 동안 명시적인 **ONTAP NFS** 마운트 옵션 세트를 구성하려면 어떻게 해야 합니까**?**

기본적으로 Trident Kubernetes에서 마운트 옵션을 어떤 값으로도 설정하지 않습니다. Kubernetes 스토리지 클래스에서 마운트 옵션을 지정하려면 다음 예를 따르세요."여기" .

프로비저닝된 볼륨을 특정 내보내기 정책으로 설정하려면 어떻게 해야 하나요**?**

적절한 호스트가 볼륨에 액세스할 수 있도록 하려면 다음을 사용하세요. `exportPolicy` 백엔드 정의 파일에 구성된 매개변수입니다.

ONTAP 사용하여 **Trident** 통해 볼륨 암호화를 설정하려면 어떻게 해야 합니까**?**

백엔드 정의 파일의 암호화 매개변수를 사용하여 Trident 에서 프로비저닝한 볼륨에 암호화를 설정할 수 있습니다. 자세한 내용은 다음을 참조하세요."Trident NVE 및 NAE와 함께 작동하는 방식"

Trident 통해 **ONTAP** 에 대한 **QoS**를 구현하는 가장 좋은 방법은 무엇입니까**?**

사용 `StorageClasses` ONTAP 에 QoS를 구현합니다.

Trident 통해 씬 프로비저닝이나 씩 프로비저닝을 지정하려면 어떻게 해야 하나요**?**

ONTAP 드라이버는 씬 프로비저닝이나 씩 프로비저닝을 모두 지원합니다. ONTAP 드라이버는 기본적으로 씬 프로비저닝을 사용합니다. 두꺼운 프로비저닝이 필요한 경우 백엔드 정의 파일이나 다음을 구성해야 합니다. `StorageClass` . 둘 다 구성된 경우, `StorageClass` 우선합니다. ONTAP 에 대해 다음을 구성합니다.

1. ~에 `StorageClass` , 설정하다 `provisioningType` 속성이 두껍습니다.
2. 백엔드 정의 파일에서 다음을 설정하여 두꺼운 볼륨을 활성화합니다. `backend spaceReserve parameter` 볼륨으로서.

실수로 **PVC**를 삭제하더라도 사용 중인 볼륨이 삭제되지 않도록 하려면 어떻게 해야 합니까**?**

PVC 보호는 Kubernetes 버전 1.10부터 자동으로 활성화됩니다.

Trident 에서 만든 **NFS PVC**를 재배할 수 있나요**?**

네. Trident 에서 생성된 PVC를 확장할 수 있습니다. 볼륨 자동 증가는 Trident 에 적용할 수 없는 ONTAP 기능입니다.

SnapMirror Data Protection(DP) 또는 오프라인 모드에 있는 동안 볼륨을 가져올 수 있나요**?**

외부 볼륨이 DP 모드이거나 오프라인인 경우 볼륨 가져오기가 실패합니다. 다음과 같은 오류 메시지가 나타납니다.

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

리소스 할당량은 **NetApp** 클러스터로 어떻게 변환되나요**?**

Kubernetes 스토리지 리소스 할당량은 NetApp 스토리지에 용량이 있는 한 작동해야 합니다. NetApp 스토리지가 용량 부족으로 인해 Kubernetes 할당량 설정을 준수할 수 없는 경우 Trident 프로비저닝을 시도하지만 오류가 발생합니다.

Trident 사용하여 볼륨 스냅샷을 만들 수 있나요**?**

네. Trident 에서는 스냅샷에서 주문형 볼륨 스냅샷과 영구 볼륨을 생성하는 기능을 지원합니다. 스냅샷에서 PV를 생성하려면 다음을 확인하세요. `VolumeSnapshotDataSource` 기능 게이트가 활성화되었습니다.

Trident 볼륨 스냅샷을 지원하는 드라이버는 무엇입니까**?**

오늘부터 주문형 스냅샷 지원이 가능합니다. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `gcp-cvs` , 그리고 `azure-netapp-files` 백엔드 드라이버.

ONTAP 사용하여 **Trident** 에서 프로비저닝한 볼륨의 스냅샷 백업을 어떻게 만들 수 있나요**?**

이것은 다음에서 사용 가능합니다. `ontap-nas` , `ontap-san` , 그리고 `ontap-nas-flexgroup` 운전자. 또한 다음을 지정할 수도 있습니다. `snapshotPolicy` 를 위해 `ontap-san-economy` FlexVol 레벨의 드라이버.

이것은 또한 다음에서도 사용 가능합니다. `ontap-nas-economy` 드라이버는 FlexVol volume 수준 세분성에 따라 동작하며 qtree 수준 세분성에는 동작하지 않습니다. Trident 에서 프로비저닝한 볼륨의 스냅샷 기능을 활성화하려면 백엔드 매개변수 옵션을 설정하세요. `snapshotPolicy` ONTAP 백엔드에 정의된 원하는 스냅샷 정책에 따라. 스토리지 컨트롤러가 찍은 스냅샷은 Trident 에서 알 수 없습니다.

Trident 통해 프로비저닝된 볼륨에 대해 스냅샷 예약 비율을 설정할 수 있나요**?**

예, Trident 를 통해 스냅샷 복사본을 저장하기 위해 디스크 공간의 특정 백분율을 예약할 수 있습니다. `snapshotReserve` 백엔드 정의 파일의 속성. 구성한 경우 `snapshotPolicy` 그리고 `snapshotReserve` 백엔드 정의 파일에서 스냅샷 예약 비율은 다음에 따라 설정됩니다. `snapshotReserve` 백엔드 파일에 언급된 백분율입니다. 만약 `snapshotReserve` 백분율 숫자는 언급되지 않았지만 ONTAP 기본적으로 스냅샷 예약 백분율을 5로 설정합니다. 만약 `snapshotPolicy` 옵션이 없음으로 설정되면 스냅샷 예약 비율이 0으로 설정됩니다.

볼륨 스냅샷 디렉토리에 직접 접근하여 파일을 복사할 수 있나요**?**

예, Trident 에서 프로비저닝한 볼륨의 스냅샷 디렉토리에 액세스하려면 다음을 설정하세요. `snapshotDir` 백엔드 정의 파일의 매개변수.

Trident 통해 볼륨에 대한 **SnapMirror** 설정할 수 있나요**?**

현재 SnapMirror ONTAP CLI 또는 OnCommand System Manager 사용하여 외부에서 설정해야 합니다.

영구 볼륨을 특정 **ONTAP** 스냅샷으로 복원하려면 어떻게 해야 하나요**?**

볼륨을 ONTAP 스냅샷으로 복원하려면 다음 단계를 수행하세요.

1. 영구 볼륨을 사용하는 애플리케이션 포드를 중지합니다.
2. ONTAP CLI 또는 OnCommand System Manager 통해 필요한 스냅샷으로 되돌립니다.
3. 애플리케이션 포드를 다시 시작합니다.

Trident 로드 공유 미러가 구성된 **SVM**에서 볼륨을 프로비저닝할 수 있나요**?**

NFS를 통해 데이터를 제공하는 SVM의 루트 볼륨에 대해 로드 공유 미러를 생성할 수 있습니다. ONTAP Trident 에서 생성된 볼륨에 대한 로드 공유 미러를 자동으로 업데이트합니다. 이로 인해 볼륨 증가가 지연될 수 있습니다. Trident 사용하여 여러 볼륨을 생성하는 경우 볼륨 프로비저닝은 ONTAP 로드 공유 미러를 업데이트하는 데 따라 달라집니다.

각 고객/테넌트에 대한 스토리지 클래스 사용량을 어떻게 분리할 수 있나요**?**

Kubernetes는 네임스페이스에서 스토리지 클래스를 허용하지 않습니다. 그러나 Kubernetes를 사용하면 네임스페이스별로 적용되는 스토리지 리소스 할당량을 사용하여 네임스페이스별로 특정 스토리지 클래스의 사용을 제한할 수 있습니다. 특정 네임스페이스가 특정 저장소에 액세스하는 것을 거부하려면 해당 저장소 클래스에 대한

리소스 할당량을 0으로 설정합니다.

문제 해결

여기에 제공된 포인터를 사용하여 Trident 설치하고 사용하는 동안 발생할 수 있는 문제를 해결하세요.

Trident 에 대한 도움이 필요하면 다음을 사용하여 지원 번들을 만드십시오. `tridentctl logs -a -n trident` NetApp 지원팀으로 보내주세요.

일반적인 문제 해결

- Trident 포드가 제대로 올라오지 않는 경우(예: Trident 포드가 끼어 있는 경우) `ContainerCreating` 2개 미만의 준비된 컨테이너가 있는 단계) 실행 중 `kubectl -n trident describe deployment trident` 그리고 `kubectl -n trident describe pod trident-**` 추가적인 통찰력을 제공할 수 있습니다. kubelet 로그 얻기(예: `journalctl -xeu kubelet`)도 도움이 될 수 있습니다.
- Trident 로그에 충분한 정보가 없으면 다음을 전달하여 Trident 에 대한 디버그 모드를 활성화해 볼 수 있습니다. `-d` 설치 옵션에 따라 설치 매개변수에 플래그를 지정합니다.

  그런 다음 디버그가 설정되었는지 확인하십시오. `./tridentctl logs -n trident` 그리고 검색 중 `level=debug msg` 로그에.

  Operator로 설치됨

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  이렇게 하면 모든 Trident 포드가 재시작되는데, 몇 초 정도 걸릴 수 있습니다. 출력에서 'AGE' 열을 관찰하여 이를 확인할 수 있습니다. `kubectl get pod -n trident` .

  Trident 20.07 및 20.10 사용 `tprov` 대신하여 `torc` .

  Helm과 함께 설치됨

  ```
  helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  tridentctl로 설치됨

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- 다음을 포함하여 각 백엔드에 대한 디버그 로그를 얻을 수도 있습니다. `debugTraceFlags` 백엔드 정의에서. 예를 들어 다음을 포함합니다. `debugTraceFlags: {"api":true, "method":true,}` Trident 로그에서 API 호출과 메서드 탐색을 얻습니다. 기존 백엔드는 다음을 가질 수 있습니다. `debugTraceFlags` 로 구성됨 `tridentctl backend update` .

- Red Hat Enterprise Linux CoreOS(RHCOS)를 사용하는 경우 다음 사항을 확인하십시오. `iscsid` 작업자 노드에서 활성화되어 있으며 기본적으로 시작됩니다. 이 작업은 OpenShift MachineConfigs를 사용하거나 점화 템플릿을 수정하여 수행할 수 있습니다.
- Trident 를 사용할 때 발생할 수 있는 일반적인 문제 "Azure NetApp Files" 테넌트와 클라이언트 비밀이 권한이 부족한 앱 등록에서 나오는 경우입니다. Trident 요구 사항의 전체 목록은 다음을 참조하세요."Azure NetApp Files" 구성.
- 컨테이너에 PV를 장착하는 데 문제가 있는 경우 다음을 확인하십시오. `rpcbind` 설치 및 실행 중입니다. 호스트 OS에 필요한 패키지 관리자를 사용하여 확인하십시오. `rpcbind` 실행 중입니다. 상태를 확인할 수 있습니다 `rpcbind` 서비스를 실행하여 `systemctl status rpcbind` 또는 이에 상응하는 것.
- Trident 백엔드가 다음 위치에 있다고 보고하는 경우 `failed` 이전에는 작동했던 상태이지만 백엔드와 관련된 SVM/관리자 자격 증명을 변경한 것으로 인해 발생한 것일 가능성이 높습니다. 백엔드 정보 업데이트 `tridentctl update backend` 또는 Trident 포드를 튀기면 이 문제가 해결됩니다.
- 컨테이너 런타임으로 Docker를 사용하여 Trident 설치할 때 권한 문제가 발생하는 경우 다음을 사용하여 Trident 를 설치해 보세요. `--in cluster=false` 깃발. 이렇게 하면 설치 프로그램 포드를 사용하지 않으므로 다음과 같은 권한 문제가 발생하지 않습니다. `trident-installer` 사용자.
- 사용하다 `uninstall parameter <Uninstalling Trident>` 실패한 실행 후 정리하기 위해. 기본적으로 스크립트는 Trident 에서 생성된 CRD를 제거하지 않으므로 실행 중인 배포에서도 안전하게 제거한 후 다시 설치할 수 있습니다.
- Trident 의 이전 버전으로 다운그레이드하려면 먼저 다음을 실행하세요. `tridentctl uninstall` Trident 제거하는 명령. 원하는 것을 다운로드하세요 "Trident 버전" 그리고 다음을 사용하여 설치하세요 `tridentctl install` 명령.
- 성공적인 설치 후 PVC가 끼어 있는 경우 `Pending` 단계, 실행 중 `kubectl describe pvc` Trident 이 PVC에 대한 PV를 프로비저닝하지 못한 이유에 대한 추가 정보를 제공할 수 있습니다.

운영자를 사용하여 **Trident** 배포가 실패했습니다.

운영자를 사용하여 Trident 배포하는 경우 상태 `TridentOrchestrator` 변경 사항 `Installing` 에게 `Installed` . 당신이 관찰한다면 `Failed` 상태가 좋지 않고 운영자가 스스로 복구할 수 없는 경우 다음 명령을 실행하여 운영자의 로그를 확인해야 합니다.

```
tridentctl logs -l trident-operator
```

트라이던트 연산자 컨테이너의 로그를 추적하면 문제가 있는 곳을 알 수 있습니다. 예를 들어, 그러한 문제 중 하나는 에어갭 환경에서 업스트림 레지스트리에서 필요한 컨테이너 이미지를 가져올 수 없다는 것입니다.

Trident 설치가 실패한 이유를 이해하려면 다음을 살펴보세요. `TridentOrchestrator` 상태.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

이 오류는 이미 존재함을 나타냅니다. `TridentOrchestrator` Trident 설치하는 데 사용되었습니다. 각 Kubernetes 클러스터에는 Trident 인스턴스가 하나만 있을 수 있으므로 운영자는 주어진 시간에 활성 인스턴스가 하나만 존재하도록 보장합니다. `TridentOrchestrator` 창조할 수 있다는 것입니다.

또한, Trident 포드의 상태를 관찰하면 무언가 잘못되었을 때를 알 수 있는 경우가 많습니다.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

컨테이너 이미지 하나 이상을 가져오지 못했기 때문에 포드가 완전히 초기화되지 않았다는 것을 명확하게 알 수 있습니다.

문제를 해결하려면 다음을 편집해야 합니다. `TridentOrchestrator` 크.알. 또는 삭제할 수 있습니다 `TridentOrchestrator` , 수정되고 정확한 정의를 사용하여 새 정의를 만듭니다.

Trident 배포 실패 `tridentctl`

무엇이 잘못되었는지 알아내려면 다음을 사용하여 설치 프로그램을 다시 실행할 수 있습니다.`-d` 디버그 모드를 켜고 문제가 무엇인지 이해하는 데 도움이 되는 인수:

```
./tridentctl install -n trident -d
```

문제를 해결한 후 다음과 같이 설치를 정리한 다음 다음을 실행할 수 있습니다. `tridentctl install` 다시 명령을 내리세요:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Trident 와 CRD를 완전히 제거하세요

Trident 와 생성된 모든 CRD 및 관련 사용자 정의 리소스를 완전히 제거할 수 있습니다.

이는 취소될 수 없습니다. Trident 를 완전히 새로 설치하려는 경우가 아니라면 이 작업을 수행하지 마세요. CRD를 제거하지 않고 Trident 제거하려면 다음을 참조하세요."Trident 제거" .

Trident 연산자

Trident 연산자를 사용하여 Trident 제거하고 CRD를 완전히 제거하려면:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

지배

Helm을 사용하여 Trident 제거하고 CRD를 완전히 제거하려면:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>트라이던트**ctl</code>**

Trident 제거한 후 CRD를 완전히 제거하려면 다음을 사용하십시오. `tridentctl`

```
tridentctl obliviate crd
```

Kubernetes 1.26에서 RWX 원시 블록 네임스페이스로 인한 NVMe 노드 언스테이징 실패

Kubernetes 1.26을 실행하는 경우 RWX 원시 블록 네임스페이스와 함께 NVMe/TCP를 사용하면 노드 스테이징 취소가 실패할 수 있습니다. 다음 시나리오는 이러한 오류에 대한 해결 방법을 제공합니다. 또는 Kubernetes를 1.27로 업그레이드할 수 있습니다.

네임스페이스와 포드를 삭제했습니다.

Pod에 Trident 관리 네임스페이스(NVMe 영구 볼륨)가 연결된 시나리오를 생각해 보세요. ONTAP 백엔드에서 네임스페이스를 직접 삭제하면 Pod를 삭제하려고 시도한 후에 스테이징 해제 프로세스가 중단됩니다. 이 시나리오는 Kubernetes 클러스터나 기타 기능에 영향을 미치지 않습니다.

해결 방법

해당 노드에서 영구 볼륨(해당 네임스페이스에 해당)을 마운트 해제하고 삭제합니다.

차단된 데이터 **LIF**

```
 If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
```
.해결 방법
모든 기능을 복원하려면 `dataLIFS`를 불러오세요.

삭제된 네임스페이스 매핑

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
```
.해결 방법
추가하세요 `hostNQN` 하위 시스템으로 돌아갑니다.

ONTAP 업그레이드 후 "v4.2-xattrs"가 활성화될 것으로 예상할 때 NFSv4.2 클라이언트가 "잘못된 인수"를 보고합니다.

ONTAP 업그레이드한 후 NFSv4.2 클라이언트가 NFSv4.2 내보내기를 마운트하려고 할 때 "잘못된 인수" 오류를 보고할 수 있습니다. 이 문제는 다음과 같은 경우에 발생합니다. `v4.2-xattrs` SVM에서는 옵션이 활성화되어 있지 않습니다. .해결 방법 활성화 `v4.2-xattrs` SVM에서 옵션을 선택하거나 ONTAP 9.12.1 이상으로 업그레이드하면 이 옵션이 기본적으로 활성화됩니다.

지원하다

NetApp 다양한 방법으로 Trident 에 대한 지원을 제공합니다. 지식 기반(KB) 문서와 Discord 채널 등 광범위한 무료 셀프 지원 옵션을 24시간 연중무휴로 이용할 수 있습니다.

Trident 지원

Trident 버전에 따라 세 가지 수준의 지원을 제공합니다. 참조하다"정의에 대한 NetApp 소프트웨어 버전 지원" .

전면적인 지원
: Trident 출시일로부터 12개월 동안 전체 지원을 제공합니다.

제한된 지원
: Trident 출시일로부터 13~24개월 동안 제한적인 지원을 제공합니다.

자립 지원
: Trident 문서는 출시일로부터 25~36개월 동안 이용 가능합니다.

버전	전면적인 지원	제한된 지원	자립 지원

"25.06"	2026년 6월	2027년 6월	2028년 6월
"25.02"	2026년 2월	2027년 2월	2028년 2월
"24.10"	2025년 10월	2026년 10월	2027년 10월
"24.06"	2025년 6월	2026년 6월	2027년 6월
"24.02"	2025년 2월	2026년 2월	2027년 2월
"23.10"	—	2025년 10월	2026년 10월
"23.07"	—	2025년 7월	2026년 7월
"23.04"	—	2025년 4월	2026년 4월
"23.01"	—	—	2026년 1월
"22.10"	—	—	2025년 10월

자립 지원

문제 해결 문서의 포괄적인 목록은 다음을 참조하세요. "NetApp 지식베이스(로그인 필요)" .

커뮤니티 지원

우리 사이트에는 컨테이너 사용자(Trident 개발자 포함)의 활발한 공개 커뮤니티가 있습니다."디스코드 채널" . 이곳은 프로젝트에 대한 일반적인 질문을 하고, 비슷한 생각을 가진 동료들과 관련 주제를 논의하기에 좋은 곳입니다.

NetApp 기술 지원

Trident 에 대한 도움이 필요하면 다음을 사용하여 지원 번들을 만드십시오. `tridentctl logs -a -n trident` 그리고 그것을 보내다 `NetApp Support <Getting Help>` .

더 많은 정보를 원하시면

- "Trident 리소스"
- "쿠버네티스 허브"

참조

Trident 포트

Trident 통신에 사용하는 포트에 대해 자세히 알아보세요.

Trident 포트

Trident Kubernetes 내부 통신에 다음 포트를 사용합니다.

포트	목적
8443	백채널 HTTPS
8001	Prometheus 메트릭 엔드포인트
8000	Trident REST 서버
17546	Trident daemonset Pod에서 사용하는 활성/준비 프로브 포트

설치 중에 활성/준비 프로브 포트를 변경할 수 있습니다. `--probe-port` 깃발. 이 포트가 워커 노드의 다른 프로세스에 의해 사용되고 있지 않은지 확인하는 것이 중요합니다.

Trident REST API

하는 동안"tridentctl 명령 및 옵션" Trident REST API와 상호 작용하는 가장 쉬운 방법이며, 원하는 경우 REST 엔드포인트를 직접 사용할 수도 있습니다.

REST API를 사용하는 경우

REST API는 Kubernetes가 아닌 배포 환경에서 Trident 독립 실행형 바이너리로 사용하는 고급 설치에 유용합니다.

더 나은 보안을 위해 Trident `REST API` Pod 내부에서 실행할 경우 기본적으로 localhost로 제한됩니다. 이 동작을 변경하려면 Trident를 설정해야 합니다. `-address` 포드 구성의 인수입니다.

REST API 사용

이러한 API가 호출되는 방법에 대한 예를 보려면 디버그를 전달하세요.(`-d`) 깃발. 자세한 내용은 다음을 참조하세요 ."tridentctl을 사용하여 Trident 관리" .

API는 다음과 같이 작동합니다.

얻다

`GET <trident-address>/trident/v1/<object-type>`

해당 유형의 모든 객체를 나열합니다.

GET <trident-address>/trident/v1/<object-type>/<object-name>

명명된 객체의 세부 정보를 가져옵니다.

우편

POST <trident-address>/trident/v1/<object-type>

지정된 유형의 객체를 생성합니다.

- 객체를 생성하려면 JSON 구성이 필요합니다. 각 객체 유형의 사양은 다음을 참조하세요."tridentctl을 사용하여 Trident 관리" .
- 해당 객체가 이미 존재하는 경우 동작이 달라집니다. 백엔드는 기존 객체를 업데이트하지만 다른 모든 객체 유형은 작업에 실패합니다.

삭제

DELETE <trident-address>/trident/v1/<object-type>/<object-name>

지정된 리소스를 삭제합니다.

> 백엔드 또는 스토리지 클래스와 연결된 볼륨은 계속 존재합니다. 이러한 볼륨은 별도로 삭제해야 합니다. 자세한 내용은 다음을 참조하세요."tridentctl을 사용하여 Trident 관리" .

명령줄 옵션

Trident Trident 오케스트레이터에 대한 여러 가지 명령줄 옵션을 제공합니다. 이러한 옵션을 사용하여 배포를 수정할 수 있습니다.

벌채 반출

-debug

디버깅 출력을 활성화합니다.

-loglevel <level>

로깅 수준(디버그, 정보, 경고, 오류, 치명적)을 설정합니다. 기본적으로 info로 설정됩니다.

쿠버네티스

-k8s_pod

이 옵션을 사용하거나 `-k8s_api_server` Kubernetes 지원을 활성화합니다. 이것을 설정하면 Trident 포함된 Pod의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 접속합니다. 이 기능은 Trident 서비스 계정이 활성화된 Kubernetes 클러스터에서 Pod로 실행되는 경우에만 작동합니다.

-k8s_api_server <insecure-address:insecure-port>

이 옵션을 사용하거나 `-k8s_pod` Kubernetes 지원을 활성화합니다. 이 옵션을 지정하면 Trident 제공된 안전하지 않은 주소와 포트를 사용하여 Kubernetes API 서버에 연결합니다. 이를 통해 Trident 포드 외부에 배포할 수 있습니다. 하지만 API 서버에 대한 안전하지 않은 연결만 지원합니다. 안전하게 연결하려면 포드에 Trident 배치하세요. `-k8s_pod` 옵션.

도커

-volume_driver <name>

Docker 플러그인을 등록할 때 사용되는 드라이버 이름입니다. 기본값으로 설정됨 `netapp` .

-driver_port <port-number>

UNIX 도메인 소켓이 아닌 이 포트에서 수신합니다.

-config <file>

필수. 백엔드 구성 파일에 대한 경로를 지정해야 합니다.

나머지

-address <ip-or-host>

Trident의 REST 서버가 수신해야 하는 주소를 지정합니다. 기본값은 localhost입니다. 로컬호스트에서 수신하고 Kubernetes Pod 내부에서 실행하는 경우 REST 인터페이스는 Pod 외부에서 직접 액세스할 수 없습니다. 사용 `-address ""` Pod IP 주소에서 REST 인터페이스에 접근할 수 있도록 합니다.

> Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신하고 서비스하도록 구성할 수 있습니다.

-port <port-number>

Trident의 REST 서버가 수신해야 하는 포트를 지정합니다. 기본값은 8000입니다.

-rest

REST 인터페이스를 활성화합니다. 기본값은 true입니다.

Kubernetes 및 Trident 객체

REST API를 사용하여 리소스 객체를 읽고 쓰는 방식으로 Kubernetes 및 Trident 와 상호 작용할 수 있습니다. Kubernetes와 Trident, Trident 와 스토리지, Kubernetes와 스토리지 간의 관계를 결정하는 여러 리소스 객체가 있습니다. 이러한 객체 중 일부는 Kubernetes를 통해 관리되고 나머지는 Trident 통해 관리됩니다.

각 객체는 서로 어떻게 상호 작용하나요?

아마도 객체를 이해하고, 객체의 용도와 상호 작용을 파악하는 가장 쉬운 방법은 Kubernetes 사용자의 단일 저장소 요청을 따르는 것입니다.

1. 사용자가 생성합니다 `PersistentVolumeClaim` 새로운 것을 요청하다 `PersistentVolume` Kubernetes에서 특정 크기의 `StorageClass` 이는 이전에 관리자가 구성한 것입니다.
2. 쿠버네티스 `StorageClass` Trident 프로비저너로 식별하고 요청된 클래스에 대한 볼륨을 프로비저닝하는 방법을 Trident 알려주는 매개변수를 포함합니다.
3. Trident 자체적으로 `StorageClass` 일치하는 것을 식별하는 동일한 이름으로 `Backends` 그리고 `StoragePools` 이를 사용하여 클래스에 대한 볼륨을 프로비저닝할 수 있습니다.
4. Trident 일치하는 백엔드에 저장소를 프로비저닝하고 두 개의 객체를 생성합니다. `PersistentVolume`

Kubernetes에서 볼륨을 찾고 마운트하고 처리하는 방법을 Kubernetes에 알려주는 볼륨과 볼륨 간의 관계를 유지하는 Trident 의 볼륨 `PersistentVolume` 그리고 실제 저장 공간.

5. Kubernetes는 다음을 바인딩합니다. `PersistentVolumeClaim` 새로운 것에 `PersistentVolume` . 다음을 포함하는 포드 `PersistentVolumeClaim` 해당 PersistentVolume을 실행 중인 모든 호스트에 마운트합니다.
6. 사용자가 생성합니다 `VolumeSnapshot` 기존 PVC를 사용하여 `VolumeSnapshotClass` Trident 를 가리키죠.
7. Trident PVC와 연관된 볼륨을 식별하고 백엔드에 볼륨의 스냅샷을 만듭니다. 또한 다음을 생성합니다. `VolumeSnapshotContent` Kubernetes에 스냅샷을 식별하는 방법을 지시합니다.
8. 사용자는 다음을 생성할 수 있습니다. `PersistentVolumeClaim` 사용 중 `VolumeSnapshot` 출처로서.
9. Trident 필요한 스냅샷을 식별하고 스냅샷 생성에 관련된 동일한 단계 세트를 수행합니다. `PersistentVolume` 그리고 `Volume` .

Kubernetes 객체에 대한 추가 정보를 읽으려면 다음을 꼭 읽어 보시기 바랍니다. "영구 볼륨" Kubernetes 문서의 섹션입니다.

쿠버네티스 `PersistentVolumeClaim` 사물

쿠버네티스 `PersistentVolumeClaim` 객체는 Kubernetes 클러스터 사용자가 만든 저장소 요청입니다.

Trident 사용하면 표준 사양 외에도 백엔드 구성에서 설정한 기본값을 재정의하려는 경우 다음과 같은 볼륨별 주석을 지정할 수 있습니다.

주석	볼륨 옵션	지원되는 드라이버
트라이던트.넷앱.io/파일시스템	파일 시스템	온탭산, 솔리드파이어산, 온탭산이코노미
트라이던트.넷앱.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economy
트라이던트.넷앱.io/splitOnClone	splitOnClone	온탑나스, 온탑산
트라이던트.넷앱.io/프로토콜	규약	어느
트라이던트.넷앱.io/내보내기정책	수출정책	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/스냅샷정책	스냅샷 정책	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산
트라이던트.넷앱.io/스냅샷예약	스냅샷예약	온탭-나스, 온탭-나스-플렉스그룹, 온탭-산, GCP-CVS
trident.netapp.io/스냅샷 디렉토리	스냅샷 디렉토리	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/유닉스권한	unixPermissions	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹
트라이던트.넷앱.io/블록사이즈	블록 크기	솔리드파이어-산

생성된 PV에 다음이 있는 경우 `Delete` 회수 정책에 따라, Trident PV가 해제되면(즉, 사용자가 PVC를 삭제하면) PV와 백업 볼륨을 모두 삭제합니다. 삭제 작업이 실패하면 Trident PV를 삭제로 표시하고 성공하거나 PV를 수동으로 삭제할 때까지 주기적으로 작업을 다시 시도합니다. PV가 사용하는 경우 `Retain` 정책에 따라 Trident 이를 무시하고

관리자가 Kubernetes와 백엔드에서 이를 정리하여 볼륨을 제거하기 전에 백업하거나 검사할 수 있다고 가정합니다. PV를 삭제해도 Trident 백업 볼륨을 삭제하지는 않습니다. REST API를 사용하여 제거해야 합니다.(`tridentctl`).

Trident CSI 사양을 사용하여 볼륨 스냅샷 생성을 지원합니다. 볼륨 스냅샷을 생성하고 이를 데이터 소스로 사용하여 기존 PVC를 복제할 수 있습니다. 이런 방식으로 PV의 특정 시점 복사본을 스냅샷 형태로 Kubernetes에 노출할 수 있습니다. 그런 다음 스냅샷을 사용하여 새로운 PV를 만들 수 있습니다. 살펴보세요 `On-Demand Volume Snapshots` 이것이 어떻게 작동하는지 알아보려고요.

Trident 또한 다음을 제공합니다. `cloneFromPVC` 그리고 `splitOnClone` 클론을 생성하기 위한 주석. 이러한 주석을 사용하면 CSI 구현을 사용하지 않고도 PVC를 복제할 수 있습니다.

다음은 예입니다. 사용자가 이미 PVC를 가지고 있는 경우 `mysql` 사용자는 새로운 PVC를 생성할 수 있습니다. `mysqlclone` 예를 들어 주석을 사용하여 `trident.netapp.io/cloneFromPVC: mysql` . 이 주석 세트를 사용하면 Trident 볼륨을 처음부터 프로비저닝하는 대신 MySQL PVC에 해당하는 볼륨을 복제합니다.

다음 사항을 고려하세요.

- NetApp 유휴 볼륨을 복제할 것을 권장합니다.
- PVC와 해당 복제본은 동일한 Kubernetes 네임스페이스에 있어야 하며 동일한 스토리지 클래스를 가져야 합니다.
- 와 함께 `ontap-nas` 그리고 `ontap-san` 드라이버의 경우 PVC 주석을 설정하는 것이 바람직할 수 있습니다. `trident.netapp.io/splitOnClone` 와 함께 `trident.netapp.io/cloneFromPVC` . 와 함께 `trident.netapp.io/splitOnClone` 로 설정 `true` Trident 복제된 볼륨을 부모 볼륨에서 분리하여 복제된 볼륨의 수명 주기를 부모 볼륨에서 완전히 분리하지만, 저장 효율성은 일부 떨어집니다. 설정하지 않음 `trident.netapp.io/splitOnClone` 또는 그것을 설정 `false` 부모 볼륨과 복제 볼륨 간에 종속성이 생성되어 복제 볼륨을 먼저 삭제하지 않으면 부모 볼륨을 삭제할 수 없게 되면서 백엔드에서 공간 사용량이 줄어듭니다. 복제본을 분할하는 것이 합리적인 시나리오 중 하나는 볼륨과 복제본이 크게 갈라지고 ONTAP 이 제공하는 저장 효율성의 이점을 얻지 못할 것으로 예상되는 빈 데이터베이스 볼륨을 복제하는 것입니다.

그만큼 `sample-input` 디렉토리에는 Trident 와 함께 사용할 수 있는 PVC 정의의 예가 포함되어 있습니다. 참조하다 Trident 볼륨과 관련된 매개변수와 설정에 대한 전체 설명을 확인하세요.

쿠버네티스 `PersistentVolume` 사물

쿠버네티스 `PersistentVolume` 객체는 Kubernetes 클러스터에서 사용할 수 있는 저장소를 나타냅니다. 포드를 사용하는 포드와 독립적인 수명 주기를 갖습니다.

> Trident 생성 `PersistentVolume` 객체를 생성하고 프로비저닝하는 볼륨에 따라 자동으로 Kubernetes 클러스터에 등록합니다. 당신이 직접 관리할 필요는 없습니다.

Trident 기반을 참조하는 PVC를 생성할 때 `StorageClass` Trident 해당 스토리지 클래스를 사용하여 새 볼륨을 프로비저닝하고 해당 볼륨에 대한 새 PV를 등록합니다. 프로비저닝된 볼륨과 해당 PV를 구성할 때 Trident 다음 규칙을 따릅니다.

- Trident Kubernetes에 대한 PV 이름과 스토리지를 프로비저닝하는 데 사용하는 내부 이름을 생성합니다. 두 경우 모두 이름이 해당 범위에서 고유하다는 점이 보장됩니다.
- 볼륨 크기는 PVC에서 요청한 크기와 최대한 일치하도록 하지만, 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있습니다.

쿠버네티스 StorageClass 사물

쿠버네티스 StorageClass 객체는 이름으로 지정됩니다. PersistentVolumeClaims 속성 집합으로 저장소를 프로비저닝합니다. 스토리지 클래스 자체는 사용될 프로비저너를 식별하고 프로비저너가 이해하는 용어로 해당 속성 집합을 정의합니다.

이는 관리자가 만들고 관리해야 하는 두 가지 기본 객체 중 하나입니다. 다른 하나는 Trident 백엔드 객체입니다.

쿠버네티스 StorageClass Trident 사용하는 객체는 다음과 같습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

이러한 매개변수는 Trident에만 적용되며 Trident 클래스에 대한 볼륨을 프로비저닝하는 방법을 알려줍니다.

저장 클래스 매개변수는 다음과 같습니다.

기인하다	유형	필수의	설명
속성	맵[문자열]문자열	아니요	아래의 속성 섹션을 참조하세요.
스토리지풀	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로
추가 스토리지 풀	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로
스토리지 풀 제외	map[문자열]문자열목록	아니요	백엔드 이름 맵을 스토리지 풀 목록으로

저장소 속성과 가능한 값은 저장소 풀 선택 속성과 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 속성

이러한 매개변수는 주어진 유형의 볼륨을 프로비저닝하는 데 어떤 Trident 관리 스토리지 풀을 사용해야 하는지 결정합니다.

기인하다	유형	가치	권하다	요구	지원됨
미디어[1]	끈	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다 의미합니다.	지정된 미디어 유형	온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-산, 솔리드파이어-산
프로비저닝 유형	끈	얇은, 두꺼운	풀은 이 프로비저닝 방법을 지원합니다.	프로비저닝 방법이 지정됨	두꺼운: 모두 온탭; 얇은: 모두 온탭 & solidfire-san
백엔드 유형	끈	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	풀은 이 유형의 백엔드에 속합니다.	백엔드 지정됨	모든 운전자
스냅샷	부울	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다.	스냅샷이 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
클론	부울	참, 거짓	풀은 볼륨 복제를 지원합니다.	복제가 활성화된 볼륨	온탭-나스, 온탭-산, 솔리드파이어-산, GCP-CVS
암호화	부울	참, 거짓	풀은 암호화된 볼륨을 지원합니다.	암호화가 활성화된 볼륨	온탑나스, 온탑나스이코노미, 온탑나스플렉스그룹, 온탑산
아이옵스	정수	양의 정수	풀은 이 범위에서 IOPS를 보장할 수 있습니다.	볼륨은 이러한 IOPS를 보장합니다.	솔리드파이어-산

[1]: ONTAP Select 시스템에서는 지원되지 않습니다.

대부분의 경우, 요청된 값은 프로비저닝에 직접적인 영향을 미칩니다. 예를 들어, 두꺼운 프로비저닝을 요청하면 두꺼운 프로비저닝된 볼륨이 생성됩니다. 하지만 Element 스토리지 풀은 요청된 값이 아닌 제공된 IOPS 최소값과 최대값을 사용하여 QoS 값을 설정합니다. 이 경우, 요청된 값은 스토리지 풀을 선택하는 데에만 사용됩니다.

이상적으로는 다음을 사용할 수 있습니다. `attributes` 특정 계층의 요구를 충족하는 데 필요한 저장소의 품질을 단독으로 모델링합니다. Trident 모든 것과 일치하는 스토리지 풀을 자동으로 검색하고 선택합니다. `attributes` 당신이 지정한 대로.

사용할 수 없는 경우 `attributes` 수업에 적합한 풀을 자동으로 선택하려면 다음을 사용할 수 있습니다. `storagePools` 그리고 `additionalStoragePools` 풀을 더욱 세분화하거나 특정 풀 세트를 선택하기 위한 매개변수입니다.

당신은 사용할 수 있습니다 `storagePools` 지정된 풀과 일치하는 풀 세트를 추가로 제한하기 위한 매개변수

`attributes` . 즉, Trident 다음에 의해 식별된 풀의 교차점을 사용합니다. `attributes` 그리고 `storagePools` 프로비저닝을 위한 매개변수. 두 매개변수 중 하나만 사용하거나 두 매개변수를 함께 사용할 수 있습니다.

당신은 사용할 수 있습니다 `additionalStoragePools` Trident 프로비저닝에 사용하는 풀 세트를 확장하기 위한 매개변수입니다. 이는 선택한 풀에 관계없이 적용됩니다. `attributes` 그리고 `storagePools` 매개변수.

당신은 사용할 수 있습니다 `excludeStoragePools` Trident 프로비저닝에 사용하는 풀 세트를 필터링하는 매개변수입니다. 이 매개변수를 사용하면 일치하는 모든 풀이 제거됩니다.

에서 `storagePools` 그리고 `additionalStoragePools` 매개변수, 각 항목은 다음 형식을 취합니다. `<backend>:<storagePoolList>` , 어디 `<storagePoolList>` 지정된 백엔드에 대한 스토리지 풀의 쉼표로 구분된 목록입니다. 예를 들어, 값 `additionalStoragePools` 처럼 보일 수도 있습니다 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze` . 이러한 목록은 백엔드 값과 목록 값 모두에 대한 정규식 값을 허용합니다. 사용할 수 있습니다 `tridentctl get backend` 백엔드와 해당 풀의 목록을 가져옵니다.

쿠버네티스 속성

이러한 속성은 동적 프로비저닝 중 Trident 가 스토리지 풀/백엔드를 선택하는 데 영향을 미치지 않습니다. 대신 이러한 속성은 Kubernetes 영구 볼륨에서 지원하는 매개변수를 제공합니다. 워커 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

기인하다	유형	가치	설명	관련 드라이버	쿠버네티스 버전
fs타입	끈	ext4, ext3, xfs	블록 볼륨의 파일 시스템 유형	솔리드파이어-샌, 온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-샌, 온탑-샌-이코노미	모두
볼륨 확장 허용	부울	참, 거짓	PVC 크기 확장 지원 활성화 또는 비활성화	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
볼륨 바인딩 모드	끈	즉시, WaitForFirstConsumer	볼륨 바인딩 및 동적 프로비저닝이 발생하는 시점을 선택하세요	모두	1.19 - 1.26

- 그만큼 `fsType` 매개변수는 SAN LUN에 대한 원하는 파일 시스템 유형을 제어하는 데 사용됩니다. 또한 Kubernetes는 다음의 존재를 사용합니다. `fsType` 파일 시스템이 존재한다는 것을 나타내기 위해 저장 클래스에 있습니다. 볼륨 소유권은 다음을 사용하여 제어할 수 있습니다. `fsGroup` 포드의 보안 컨텍스트는 다음과 같은 경우에만 해당됩니다. `fsType` 설정되었습니다. 참조하다"Kubernetes: Pod 또는 컨테이너에 대한 보안 컨텍스트 구성" 볼륨 소유권 설정에 대한 개요는 다음을 참조하세요. `fsGroup` 문맥. Kubernetes는 다음을 적용합니다. `fsGroup` 값은 다음의 경우에만 적용됩니다.

  - `fsType`저장 클래스에 설정됩니다.
  - PVC 접근 모드는 RWO입니다.

  NFS 스토리지 드라이버의 경우 파일 시스템은 이미 NFS 내보내기의 일부로 존재합니다. 사용하기 위해 `fsGroup` 저장 클래스는 여전히 다음을 지정해야 합니다. `fsType` . 설정할 수 있습니다 `nfs` 또는 null이 아닌 값.

- 참조하다"볼륨 확장" 볼륨 확장에 대한 자세한 내용은 다음을 참조하세요.
- Trident 설치 프로그램 번들은 Trident 와 함께 사용할 수 있는 여러 가지 예시 스토리지 클래스 정의를 제공합니다.`sample-input/storage-class-*.yaml` . Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래스도 삭제됩니다.

쿠버네티스 `VolumeSnapshotClass` 사물

쿠버네티스 `VolumeSnapshotClass` 객체는 다음과 유사합니다 `StorageClasses` . 이러한 스냅샷은 여러 스토리지 클래스를 정의하는 데 도움이 되며 볼륨 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 스냅샷은 단일 볼륨 스냅샷 클래스와 연결됩니다.

에이 `VolumeSnapshotClass` 스냅샷을 생성하려면 관리자가 정의해야 합니다. 다음 정의를 사용하여 볼륨 스냅샷 클래스가 생성됩니다.

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

그만큼 `driver` Kubernetes에 볼륨 스냅샷에 대한 요청을 지정합니다. `csi-snapclass` 클래스는 Trident 에 의해 처리됩니다. 그만큼 `deletionPolicy` 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. 언제 `deletionPolicy` 로 설정됩니다 `Delete` 스냅샷이 삭제되면 볼륨 스냅샷 개체와 스토리지 클러스터의 기본 스냅샷도 제거됩니다. 또는 다음과 같이 설정합니다. `Retain` 즉, `VolumeSnapshotContent` 그리고 물리적 스냅샷은 보존됩니다.

쿠버네티스 `VolumeSnapshot` 사물

쿠버네티스 `VolumeSnapshot` 객체는 볼륨의 스냅샷을 생성하라는 요청입니다. PVC가 사용자가 볼륨에 대해 한 요청을 나타내는 것처럼 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

볼륨 스냅샷 요청이 들어오면 Trident 백엔드에서 볼륨에 대한 스냅샷 생성을 자동으로 관리하고 고유한 스냅샷을 생성하여 스냅샷을 노출합니다.

VolumeSnapshotContent 물체. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

> VolumeSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident 이 PVC의 백업 볼륨을 삭제 상태로 표시하지만 완전히 제거하지는 않습니다. 연관된 모든 스냅샷이 삭제되면 볼륨이 제거됩니다.

쿠버네티스 VolumeSnapshotContent 사물

쿠버네티스 VolumeSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 가져온 스냅샷을 나타냅니다. 이것은 다음과 유사합니다. PersistentVolume 스토리지 클러스터에 프로비저닝된 스냅샷을 나타냅니다. 와 유사하다 PersistentVolumeClaim 그리고 PersistentVolume 스냅샷이 생성되면 개체 VolumeSnapshotContent 객체는 일대일 매핑을 유지합니다. VolumeSnapshot 스냅샷 생성을 요청한 객체입니다.

그만큼 VolumeSnapshotContent 개체에는 스냅샷을 고유하게 식별하는 세부 정보(예: snapshotHandle . 이것 snapshotHandle PV 이름과 VolumeSnapshotContent 물체.

스냅샷 요청이 들어오면 Trident 백엔드에서 스냅샷을 생성합니다. 스냅샷이 생성된 후 Trident 다음을 구성합니다. VolumeSnapshotContent 객체를 생성하여 Kubernetes API에 스냅샷을 노출합니다.

> 일반적으로 관리할 필요가 없습니다. VolumeSnapshotContent 물체. 이에 대한 예외는 다음을 원할 때입니다."볼륨 스냅샷 가져오기" Trident 외부에서 생성됨.

쿠버네티스 VolumeGroupSnapshotClass 사물

쿠버네티스 VolumeGroupSnapshotClass 객체는 다음과 유사합니다 VolumeSnapshotClass . 이러한 스냅샷은 여러 스토리지 클래스를 정의하는 데 도움이 되며 볼륨 그룹 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 그룹 스냅샷은 단일 볼륨 그룹 스냅샷 클래스와 연결됩니다.

에이 VolumeGroupSnapshotClass 스냅샷 그룹을 생성하려면 관리자가 정의해야 합니다. 다음 정의를 사용하여 볼륨 그룹 스냅샷 클래스가 생성됩니다.

```
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
```

그만큼 driver 볼륨 그룹 스냅샷에 대한 요청을 Kubernetes에 지정합니다. csi-group-snap-class 클래스는 Trident 에 의해 처리됩니다. 그만큼 deletionPolicy 그룹 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. 언제 deletionPolicy 로 설정됩니다 Delete 스냅샷이 삭제되면 볼륨 그룹 스냅샷 개체와 스토리지 클러스터의 기본 스냅샷도 제거됩니다. 또는 다음과 같이 설정합니다. Retain 즉, VolumeGroupSnapshotContent 그리고 물리적 스냅샷은 보존됩니다.

쿠버네티스 `VolumeGroupSnapshot` 사물

쿠버네티스 `VolumeGroupSnapshot` 객체는 여러 볼륨의 스냅샷을 생성하라는 요청입니다. PVC가 사용자가 볼륨에 대해 한 요청을 나타내는 것처럼 볼륨 그룹 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

볼륨 그룹 스냅샷 요청이 들어오면 Trident 백엔드의 볼륨에 대한 그룹 스냅샷 생성을 자동으로 관리하고 고유한 스냅샷을 생성하여 스냅샷을 노출합니다. `VolumeGroupSnapshotContent` 물체. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

> VolumeGroupSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident 이 PVC의 백업 볼륨을 삭제 상태로 표시하지만 완전히 제거하지는 않습니다. 연관된 모든 스냅샷이 삭제되면 볼륨 그룹 스냅샷도 제거됩니다.

쿠버네티스 `VolumeGroupSnapshotContent` 사물

쿠버네티스 `VolumeGroupSnapshotContent` 개체는 이미 프로비저닝된 볼륨에서 가져온 그룹 스냅샷을 나타냅니다. 이것은 다음과 유사합니다. `PersistentVolume` 스토리지 클러스터에 프로비저닝된 스냅샷을 나타냅니다. 와 유사하다 `PersistentVolumeClaim` 그리고 `PersistentVolume` 스냅샷이 생성되면 개체 `VolumeSnapshotContent` 객체는 일대일 매핑을 유지합니다. `VolumeSnapshot` 스냅샷 생성을 요청한 객체입니다.

그만큼 `VolumeGroupSnapshotContent` 개체에는 스냅샷 그룹을 식별하는 세부 정보(예: `volumeGroupSnapshotHandle` 그리고 개별 볼륨 스냅샷 핸들이 스토리지 시스템에 존재합니다.

스냅샷 요청이 들어오면 Trident 백엔드에 볼륨 그룹 스냅샷을 생성합니다. 볼륨 그룹 스냅샷이 생성된 후 Trident 다음을 구성합니다. `VolumeGroupSnapshotContent` 객체를 생성하여 Kubernetes API에 스냅샷을 노출합니다.

쿠버네티스 `CustomResourceDefinition` 사물

Kubernetes 사용자 정의 리소스는 관리자가 정의한 Kubernetes API의 엔드포인트이며 유사한 객체를 그룹화하는 데 사용됩니다. Kubernetes는 객체 컬렉션을 저장하기 위한 사용자 정의 리소스 생성을 지원합니다. 다음 리소스 정의는 다음을 실행하여 얻을 수 있습니다. `kubectl get crds` .

사용자 정의 리소스 정의(CRD)와 관련 개체 메타데이터는 Kubernetes의 메타데이터 저장소에 저장됩니다. 이로 인해 Trident 위한 별도의 매장이 필요 없게 되었습니다.

Trident 사용 `CustomResourceDefinition` Trident 백엔드, Trident 스토리지 클래스, Trident 볼륨 등 Trident 객체의 ID를 보존하는 객체입니다. 이러한 객체는 Trident 에서 관리합니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 도입합니다.

CRD는 Kubernetes 구성 요소입니다. 위에 정의된 리소스의 객체는 Trident 에 의해 생성됩니다. 간단한 예로, 백엔드가 생성될 때 `tridentctl` , 해당 `tridentbackends` CRD 객체는 Kubernetes에서 사용하기 위해 생성됩니다.

Trident의 CRD에 대해 염두에 두어야 할 몇 가지 사항은 다음과 같습니다.

- Trident 설치하면 CRD 세트가 생성되어 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- 다음을 사용하여 Trident 제거할 때 `tridentctl uninstall` 명령을 실행하면 Trident 포드가 삭제되지만 생성된 CRD는 정리되지 않습니다. 참조하다"Trident 제거" Trident 완전히 제거하고 처음부터 재구성하는 방법을 알아보세요.

Trident `StorageClass` 사물

Trident Kubernetes에 맞는 스토리지 클래스를 생성합니다. `StorageClass` 지정하는 객체 `csi.trident.netapp.io` 해당 프로비저닝 필드에 있습니다. 스토리지 클래스 이름은 Kubernetes의 이름과 일치합니다. `StorageClass` 그것이 나타내는 대상.

> Kubernetes를 사용하면 이러한 객체는 Kubernetes가 자동으로 생성됩니다. `StorageClass` Trident 프로비저너로 사용하는 것이 등록되었습니다.

저장 클래스는 볼륨에 대한 요구 사항 집합으로 구성됩니다. Trident 이러한 요구 사항을 각 스토리지 풀에 있는 속성과 일치시킵니다. 속성이 일치하면 해당 스토리지 풀은 해당 스토리지 클래스를 사용하여 볼륨을 프로비저닝하기 위한 유효한 대상이 됩니다.

REST API를 사용하여 스토리지 클래스를 직접 정의하기 위해 스토리지 클래스 구성을 생성할 수 있습니다. 그러나 Kubernetes 배포의 경우 새 Kubernetes를 등록할 때 생성될 것으로 예상합니다. `StorageClass` 사물.

Trident 백엔드 객체

백엔드는 Trident 가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스는 아무리 많은 백엔드라도 관리할 수 있습니다.

> 이는 사용자가 직접 만들고 관리하는 두 가지 객체 유형 중 하나입니다. 다른 하나는 쿠버네티스입니다 `StorageClass` 물체.

이러한 객체를 구성하는 방법에 대한 자세한 내용은 다음을 참조하세요."백엔드 구성" .

Trident `StoragePool` 사물

스토리지 풀은 각 백엔드에서 프로비저닝에 사용할 수 있는 고유한 위치를 나타냅니다. ONTAP 의 경우 이는 SVM의 집계에 해당합니다. NetApp HCI/ SolidFire 의 경우 이는 관리자가 지정한 QoS 대역에 해당합니다. Cloud Volumes Service 의 경우 이는 클라우드 공급자 지역에 해당합니다. 각 스토리지 풀에는 성능 특성과 데이터 보호 특성을 정의하는 일련의 고유한 스토리지 속성이 있습니다.

여기의 다른 객체와 달리 스토리지 풀 후보는 항상 자동으로 검색되고 관리됩니다.

Trident `Volume` 사물

볼륨은 NFS 공유, iSCSI 및 FC LUN과 같은 백엔드 엔드포인트를 포함하는 기본 프로비저닝 단위입니다. Kubernetes에서는 이것이 직접적으로 대응합니다. `PersistentVolumes` . 볼륨을 생성할 때는 볼륨을 프로비저닝할 수 있는 위치와 크기를 결정하는 스토리지 클래스가 있는지 확인하세요.

> - Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident 제공한 내용을 확인할 수 있습니다.
> - 연관된 스냅샷이 있는 PV를 삭제하면 해당 Trident 볼륨이 삭제 상태로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨이 가져야 하는 속성을 정의합니다.

기인하다	유형	필수의	설명
버전	끈	아니요	Trident API 버전("1")
이름	끈	예	생성할 볼륨의 이름
스토리지클래스	끈	예	볼륨을 프로비저닝할 때 사용할 스토리지 클래스
크기	끈	예	프로비저닝할 볼륨의 크기(바이트)
규약	끈	아니요	사용할 프로토콜 유형: "파일" 또는 "블록"
내부 이름	끈	아니요	저장 시스템의 객체 이름입니다. Trident 에서 생성됨
cloneSourceVolume	끈	아니요	ontap(nas, san) & solidfire-*: 복제할 볼륨의 이름
splitOnClone	끈	아니요	ontap(nas, san): 복제본을 부모로부터 분리합니다.
스냅샷 정책	끈	아니요	ontap-*: 사용할 스냅샷 정책
스냅샷예약	끈	아니요	ontap-*: 스냅샷을 위해 예약된 볼륨의 백분율
수출정책	끈	아니요	ontap-nas*: 사용할 정책 내보내기
스냅샷 디렉토리	부울	아니요	ontap-nas*: 스냅샷 디렉토리가 표시되는지 여부
unixPermissions	끈	아니요	ontap-nas*: 초기 UNIX 권한
블록 크기	끈	아니요	solidfire-*: 블록/섹터 크기
파일 시스템	끈	아니요	파일 시스템 유형

Trident 생성 `internalName` 볼륨을 생성할 때. 이는 두 단계로 구성됩니다. 첫째, 저장 접두사(기본값)를 추가합니다. `trident` 또는 백엔드 구성의 접두사)를 볼륨 이름에 추가하여 다음과 같은 형식의 이름을 생성합니다. `<prefix>-<volume-name>` . 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 정리합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 바꿉니다(따라서 내부 이름은 다음과 같습니다. `<prefix>_<volume-name>`). Element 백엔드의 경우 밑줄을 하이픈으로 바꿉니다.

REST API를 사용하여 볼륨 구성을 사용하여 볼륨을 직접 프로비저닝할 수 있지만 Kubernetes 배포에서는 대부분의 사용자가 표준 Kubernetes를 사용할 것으로 예상합니다. `PersistentVolumeClaim` 방법. Trident 프로비저닝 프로세스의 일부로 이 볼륨 객체를 자동으로 생성합니다.

Trident `Snapshot` 사물

스냅샷은 볼륨의 특정 시점 복사본으로, 새로운 볼륨을 프로비저닝하거나 상태를 복원하는 데 사용할 수 있습니다. Kubernetes에서는 이것이 직접적으로 대응합니다. `VolumeSnapshotContent` 사물. 각 스냅샷은 볼륨과 연결되며, 볼륨은 스냅샷의 데이터 소스입니다.

각 `Snapshot` 개체에는 아래 나열된 속성이 포함됩니다.

기인하다	유형	필수의	설명
버전	끈	예	Trident API 버전("1")
이름	끈	예	Trident 스냅샷 객체의 이름
내부 이름	끈	예	스토리지 시스템의 Trident 스냅샷 개체 이름
볼륨 이름	끈	예	스냅샷이 생성되는 영구 볼륨의 이름
볼륨 내부 이름	끈	예	스토리지 시스템의 연관된 Trident 볼륨 개체의 이름

Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident 제공한 내용을 확인할 수 있습니다.

쿠버네티스가 `VolumeSnapshot` 객체 요청이 생성되면 Trident 백업 스토리지 시스템에 스냅샷 객체를 생성하여 작동합니다. 그만큼 `internalName` 이 스냅샷 객체의 접두사를 결합하여 생성됩니다. `snapshot-` 와 함께 `UID` 의 `VolumeSnapshot` 객체(예를 들어, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` 그리고 `volumeInternalName` 백업 볼륨의 세부 정보를 얻어서 채워집니다.

Trident `ResourceQuota` 물체

Trident 데몬셋은 다음을 소모합니다. `system-node-critical` 우선순위 클래스는 Kubernetes에서 사용할 수 있는 가장 높은 우선순위 클래스로, Trident 가 노드를 정상적으로 종료하는 동안 볼륨을 식별하고 정리할 수 있도록 하며, Trident 데몬셋 포드가 리소스 압박이 높은 클러스터에서 우선순위가 낮은 워크로드를 우선적으로 처리할 수 있도록 합니다.

이를 달성하기 위해 Trident 다음을 사용합니다. `ResourceQuota` Trident 데몬셋에서 "시스템 노드에 중요한" 우선순위 클래스가 충족되는지 확인하기 위한 객체입니다. 배포 및 데몬셋 생성 전에 Trident 다음을 찾습니다. `ResourceQuota` 객체를 찾고, 발견되지 않으면 이를 적용합니다.

기본 리소스 할당량 및 우선 순위 클래스에 대한 더 많은 제어가 필요한 경우 다음을 생성할 수 있습니다. `custom.yaml` 또는 구성 `ResourceQuota` Helm 차트를 사용하여 객체를 생성합니다.

다음은 Trident 데몬셋의 우선순위를 지정하는 `ResourceQuota` 객체의 예입니다.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

리소스 할당량에 대한 자세한 내용은 다음을 참조하세요."쿠버네티스: 리소스 할당량" .

정리하다 `ResourceQuota` 설치가 실패하면

설치가 실패하는 드문 경우 `ResourceQuota` 객체가 생성되었습니다. 첫 번째 시도"설치 제거" 그리고 다시 설치하세요.

그래도 작동하지 않으면 수동으로 제거하세요. `ResourceQuota` 물체.

제거하다 `ResourceQuota`

자신의 리소스 할당을 제어하려는 경우 Trident 를 제거할 수 있습니다. `ResourceQuota` 다음 명령을 사용하여 객체를 생성합니다.

```
kubectl delete quota trident-csi -n trident
```

Pod 보안 표준(PSS) 및 보안 컨텍스트 제약 조건(SCC)

Kubernetes Pod 보안 표준(PSS)과 Pod 보안 정책(PSP)은 권한 수준을 정의하고 Pod의 동작을 제한합니다. OpenShift 보안 컨텍스트 제약 조건(SCC)은 마찬가지로 OpenShift Kubernetes Engine에 특정한 포드 제한을 정의합니다. 이러한 사용자 정의 기능을 제공하기 위해 Trident 설치 중에 특정 권한을 부여합니다. 다음 섹션에서는 Trident 가 설정한 권한에 대해 자세히 설명합니다.

PSS는 Pod 보안 정책(PSP)을 대체합니다. PSP는 Kubernetes v1.21에서 더 이상 지원되지 않으며 v1.25에서 제거될 예정입니다. 자세한 내용은 다음을 참조하세요."쿠버네티스: 보안" .

필수 Kubernetes 보안 컨텍스트 및 관련 필드

허가	설명
특권	CSI에서는 마운트 지점이 양방향이어야 합니다. 즉, Trident 노드 포드는 권한이 있는 컨테이너를 실행해야 합니다. 자세한 내용은 다음을 참조하세요."Kubernetes: 마운트 전파" .
호스트 네트워킹	iSCSI 데몬에 필요합니다. `iscsiadm` iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 NFSD와 통신하기 위해 프로세스 간 통신(IPC)을 사용합니다.
호스트 PID	시작하려면 필요합니다 `rpc-statd` NFS용. Trident 호스트 프로세스를 쿼리하여 다음을 확인합니다. `rpc-statd` NFS 볼륨을 마운트하기 전에 실행 중입니다.
역량	그만큼 `SYS_ADMIN` 기능은 권한이 있는 컨테이너의 기본 기능의 일부로 제공됩니다. 예를 들어, Docker는 권한이 있는 컨테이너에 대해 다음과 같은 기능을 설정합니다. `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
세콤프	Seccomp 프로필은 특권 컨테이너에서는 항상 "제한 없음" 상태이므로 Trident 에서 활성화할 수 없습니다.
셀리눅스	OpenShift에서는 권한이 있는 컨테이너가 실행됩니다. `spc_t` ("Super Privileged Container") 도메인에서 실행되고 권한이 없는 컨테이너는 `container_t` 도메인. ~에 `containerd` , 와 함께 `container-selinux` 설치되면 모든 컨테이너가 실행됩니다. `spc_t` SELinux를 효과적으로 비활성화하는 도메인입니다. 따라서 Trident 추가하지 않습니다. `seLinuxOptions` 컨테이너로.
DAC	권한이 있는 컨테이너는 루트로 실행해야 합니다. 권한이 없는 컨테이너는 CSI에 필요한 유닉스 소켓에 액세스하기 위해 루트로 실행됩니다.

포드 보안 표준(PSS)

상표	설명	기본
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Trident Controller와 노드가 설치 네임스페이스에 들어갈 수 있도록 허용합니다. 네임스페이스 레이블을 변경하지 마세요.	`enforce: privileged` `enforce-version: <version of the current cluster or highest version of PSS tested.>`

네임스페이스 레이블을 변경하면 포드가 예약되지 않거나, "생성 오류: …" 또는 "경고: trident-csi-…"가 발생할 수 있습니다. 이런 일이 발생하면 네임스페이스 레이블을 확인하십시오. `privileged` 변경되었습니다. 그렇다면 Trident 다시 설치하세요.

Pod 보안 정책(PSP)

필드	설명	기본
`allowPrivilegeEscalation`	권한이 있는 컨테이너는 권한 상승을 허용해야 합니다.	`true`
`allowedCSIDrivers`	Trident 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있는
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 설정보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너는 가능한 모든 기능을 부여받습니다.	비어 있는
`allowedFlexVolumes`	Trident 는 사용하지 않습니다"FlexVolume 드라이버" 따라서 허용되는 볼륨 목록에 포함되지 않습니다.	비어 있는
`allowedHostPaths`	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정하는 데에는 이점이 없습니다.	비어 있는
`allowedProcMountTypes`	Trident 어떤 것도 사용하지 않습니다 `ProcMountTypes` .	비어 있는
`allowedUnsafeSysctls`	Trident 안전하지 않은 것을 요구하지 않습니다. `sysctls` .	비어 있는
`defaultAddCapabilities`	특권 컨테이너에는 기능을 추가할 필요가 없습니다.	비어 있는
`defaultAllowPrivilegeEscalation`	권한 확대 허용은 각 Trident Pod에서 처리됩니다.	`false`
`forbiddenSysctls`	아니요 `sysctls` 허용됩니다.	비어 있는
`fsGroup`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`hostIPC`	NFS 볼륨을 마운트하려면 호스트 IPC가 통신해야 합니다. `nfsd`	`true`
`hostNetwork`	iscsiadm은 호스트 네트워크가 iSCSI 데몬과 통신하는 데 필요합니다.	`true`
`hostPID`	호스트 PID는 다음을 확인하는 데 필요합니다. `rpc-statd` 노드에서 실행 중입니다.	`true`
`hostPorts`	Trident 호스트 포트를 사용하지 않습니다.	비어 있는
`privileged`	Trident 노드 포드는 볼륨을 마운트하기 위해 권한이 있는 컨테이너를 실행해야 합니다.	`true`
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`

필드	설명	기본
requiredDropCapabilities	Trident 노드 포드는 권한이 있는 컨테이너를 실행하며 기능을 삭제할 수 없습니다.	none
runAsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
runAsUser	Trident 컨테이너는 루트로 실행됩니다.	runAsAny
runtimeClass	Trident 사용하지 않습니다 RuntimeClasses .	비어 있는
seLinux	Trident 설정되지 않습니다 seLinuxOptions 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문입니다.	비어 있는
supplementalGroups	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
volumes	Trident 포드에는 이러한 볼륨 플러그인이 필요합니다.	hostPath, projected, emptyDir

보안 컨텍스트 제약 조건(SCC)

라벨	설명	기본
allowHostDirVolumePlugin	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트합니다.	true
allowHostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 통신해야 합니다. nfsd .	true
allowHostNetwork	iscsiadm은 호스트 네트워크가 iSCSI 데몬과 통신하는 데 필요합니다.	true
allowHostPID	호스트 PID는 다음을 확인하는 데 필요합니다. rpc-statd 노드에서 실행 중입니다.	true
allowHostPorts	Trident 호스트 포트를 사용하지 않습니다.	false
allowPrivilegeEscalation	권한이 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowPrivilegedContainer	Trident 노드 포드는 볼륨을 마운트하기 위해 권한이 있는 컨테이너를 실행해야 합니다.	true
allowedUnsafeSysctls	Trident 안전하지 않은 것을 요구하지 않습니다. sysctls .	none

라벨	설명	기본
`allowedCapabilities`	권한이 없는 Trident 컨테이너는 기본 설정보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너는 가능한 모든 기능을 부여받습니다.	비어 있는
`defaultAddCapabilities`	특권 컨테이너에는 기능을 추가할 필요가 없습니다.	비어 있는
`fsGroup`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`groups`	이 SCC는 Trident 에만 적용되며 사용자에게 적용됩니다.	비어 있는
`readOnlyRootFilesystem`	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 포드는 권한이 있는 컨테이너를 실행하며 기능을 삭제할 수 없습니다.	`none`
`runAsUser`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`seLinuxContext`	Trident 설정되지 않습니다 `seLinuxOptions` 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문입니다.	비어 있는
`seccompProfiles`	특권 컨테이너는 항상 "제한되지 않음"으로 실행됩니다.	비어 있는
`supplementalGroups`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`users`	Trident 네임스페이스에서 이 SCC를 Trident 사용자에 바인딩하기 위해 하나의 항목이 제공됩니다.	해당 없음
`volumes`	Trident 포드에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, downwardAPI, projected, emptyDir`

법적 고지 사항

법적 고지사항은 저작권 표시, 상표, 특허 등에 대한 정보를 제공합니다.

저작권

"https://www.netapp.com/company/legal/copyright/"

상표

NETAPP, NETAPP 로고 및 NetApp 상표 페이지에 나열된 마크는 NetApp, Inc.의 상표입니다. 다른 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.

"https://www.netapp.com/company/legal/trademarks/"

특허

NetApp 이 소유한 현재 특허 목록은 다음에서 확인할 수 있습니다.

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

개인정보 보호정책

"https://www.netapp.com/company/legal/privacy-policy/"

오픈소스

Trident 용 NetApp 소프트웨어에 사용된 타사 저작권 및 라이센스는 각 릴리스의 공지 파일에서 검토할 수 있습니다. https://github.com/NetApp/trident/ .