Trident 25.10 문서

Trident

NetApp
February 25, 2026

목차

Trident 25.10 문서

릴리스 노트

새로운 기능

릴리스 노트는 최신 버전의 NetApp Trident에 포함된 새로운 기능, 개선 사항 및 버그 수정에 대한 정보를 제공합니다.

> 설치 프로그램 zip 파일에 포함된 Linux용 `tridentctl` 바이너리는 테스트 및 지원되는 버전입니다. zip 파일의 `/extras` 부분에 제공된 `macos` 바이너리는 테스트되거나 지원되지 않음을 유의하십시오.

25.10의 새로운 기능

개선 사항, 수정 사항 및 사용 중단을 포함한 Trident 및 Trident Protect의 새로운 기능에 대해 알아보십시오.

Trident

향상된 기능

- **Kubernetes**:
  - ONTAP-SAN(iSCSI 및 FC) 외에 ONTAP-NAS NFS 및 ONTAP-SAN-Economy 드라이버용 v1beta1 Volume Group Snapshot Kubernetes API로 CSI Volume Group Snapshot에 대한 지원이 추가되었습니다. "볼륨 그룹 스냅샷 작업"을 참조하십시오.
  - ONTAP-NAS 및 ONTAP-NAS-Economy(두 NAS 드라이버 모두에서 SMB 제외), ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버에 대해 강제 볼륨 분리를 통한 자동 워크로드 페일오버 지원이 추가되었습니다. "Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화"을 참조하십시오.
  - FCP 볼륨에 대한 노드 작업의 확장성을 높이기 위해 Trident 노드 동시 처리 기능이 향상되었습니다.
  - ONTAP NAS 드라이버에 ONTAP AFX 지원이 추가되었습니다. "ONTAP NAS 구성 옵션 및 예"를 참조하십시오.
  - TridentOrchestrator CR 및 Helm 차트 값을 통해 Trident 컨테이너의 CPU 및 메모리 리소스 요청 및 제한을 구성하는 지원이 추가되었습니다. ("문제 #1000", "문제 #927", "문제 #853", "문제 #592", "문제 #110").
  - ASAr2 personality에 대한 FC 지원이 추가되었습니다. "ONTAP SAN 구성 옵션 및 예"을 참조하십시오.
  - HTTP 대신 HTTPS로 Prometheus 메트릭을 제공하는 옵션을 추가했습니다. "Trident 모니터링"를 참조하십시오.
  - 볼륨을 가져올 때 원래 이름을 유지하면서 Trident가 볼륨의 수명 주기를 관리하도록 하는 옵션을 추가했습니다. `--no-rename` "볼륨 가져오기"을 참조하십시오.
  - Trident 배포는 이제 system-cluster-critical 우선순위 클래스에서 실행됩니다.
- helm, operator 및 tridentctl("문제 #858")을 통해 Trident 컨트롤러가 호스트 네트워킹을 사용할 수 있는 옵션이 추가되었습니다.
- Trident 25.10에서 ANF 드라이버에 수동 QoS 지원이 추가되어 프로덕션 환경에서 사용할 수 있게 되었으며, 이 실험적 향상 기능은 Trident 25.06에서 도입되었습니다.

실험적 개선 사항

운영 환경에서는 사용할 수 없습니다.

- **[기술 미리보기]**: 기존 ONTAP-SAN 드라이버(통합 ONTAP 9의 iSCSI 및 FCP 프로토콜)에 대한 기술 미리보기에 더해 ONTAP-NAS(NFS만 해당) 및 ONTAP-SAN(통합 ONTAP 9용 NVMe)에 대한 동시성 지원이 추가되었습니다.

수정 사항

- **Kubernetes**:
  - Linux DaemonSet을 node-driver-registrar로 표준화하여 Windows DaemonSet 및 컨테이너 이미지 명명 규칙과 일치하도록 CSI node-driver-registrar 컨테이너 이름 불일치를 수정했습니다.
  - 기존 qtree에 대한 엑스포트 정책이 제대로 업그레이드되지 않던 문제를 해결했습니다.
- **Openshift**:
  - Openshift의 Windows 노드에서 SCC의 allowHostDirVolumePlugin이 false로 설정되어 있어 Trident 노드 pod가 시작되지 않던 문제를 수정했습니다("문제 #950").
- Helm을 통해 Kubernetes API QPS가 설정되지 않던 문제를 수정했습니다("문제 #975").
- 동일한 Kubernetes 노드에서 NVMe 기반 XFS 파일 시스템 PVC의 스냅샷을 기반으로 영구 볼륨 클레임(PVC)을 마운트할 수 없었던 문제를 수정했습니다.
- NDVP 모드에서 호스트/Docker 재시작 후 발생하는 UUID 변경 문제를 해결하기 위해 백엔드별로 고유/공유 하위 시스템 이름(예: netappdvp_subsystem)을 추가했습니다.
- Trident 23.10 이전 버전에서 24.10 이상 버전으로 업그레이드하는 동안 iSCSI 볼륨 마운트 오류를 수정하여 "invalid SANType" 문제를 해결했습니다.
- Trident 컨트롤러를 재시작하지 않으면 Trident 백엔드 상태가 온라인/오프라인으로 전환되지 않던 문제를 수정했습니다.
- 느린 PVC 크기 조정을 유발하는 간헐적인 경합 조건을 수정했습니다.
- 볼륨 클론 실패 시 스냅샷이 정리되지 않던 문제를 해결했습니다.
- 커널에 의해 장치 경로가 변경될 때 볼륨을 스테이징 해제하지 못하는 오류를 수정했습니다.
- LUKS 장치가 이미 닫혀 있어서 볼륨을 스테이징 해제할 수 없었던 문제를 수정했습니다.
- 저장 작업 속도가 느려 ContextDeadline 오류가 발생하던 문제를 수정했습니다.
- Trident Operator는 설정 가능한 k8s-timeout 동안 대기하여 Trident 버전을 확인합니다.

Trident Protect

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너를 기반으로 하는 스테이트풀 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다.

향상된 기능

- 스케줄 및 백업 CR에 대한 스냅샷 CR 시간 초과를 제어하기 위한 주석을 추가했습니다.
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`

- `protect.trident.netapp.io/volume-snapshots-created-timeout`

  "지원되는 백업 및 일정 주석"을 참조하십시오.

- 백업 CR에서 사용될 PVC 바인딩 시간 초과를 구성하기 위해 스케줄 CR에 주석을 추가했습니다 `protect.trident.netapp.io/pvc-bind-timeout-sec`. "지원되는 백업 및 일정 주석"을 참조하십시오.
- 개선된 `tridentctl-protect` 백업 및 스냅샷 목록에 실행 후크 실패를 나타내는 새 필드가 추가되었습니다.

25.06.2의 변경 사항

Trident

수정 사항

- **Kubernetes**: Kubernetes 노드에서 볼륨을 분리할 때 잘못된 iSCSI 장치가 검색되는 심각한 문제를 수정했습니다.

25.06.1의 변경 사항

Trident

> SolidFire를 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

- **Kubernetes**:
  - 하위 시스템에서 매핑 해제되기 전에 NQN이 확인되지 않던 문제를 해결했습니다.
  - LUKS 장치를 여러 번 닫으려고 시도할 경우 볼륨 분리가 실패하는 문제를 수정했습니다.
  - iSCSI 볼륨이 생성된 이후 디바이스 경로가 변경되었을 때 볼륨 스테이징 해제 문제를 해결했습니다.
  - 스토리지 클래스 간 볼륨의 블록 복제.
- **OpenShift**: OCP 4.19에서 iSCSI 노드 준비가 실패하는 문제를 수정했습니다.
- SolidFire 백엔드를 사용하여 볼륨 클론을 생성할 때 타임아웃 시간을 늘렸습니다("문제 #1008").

25.06의 변경 사항

Trident

향상된 기능

- **Kubernetes**:
  - `v1beta1` Volume Group Snapshot Kubernetes API를 사용하여 ONTAP-SAN iSCSI 드라이버용 CSI Volume Group Snapshot 지원이 추가되었습니다. "볼륨 그룹 스냅샷 작업"을 참조하십시오.

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

- iSCSI 외에도 NVMe/TCP용 ONTAP ASA r2 지원이 추가되었습니다. "ONTAP SAN 구성 옵션 및 예"을 참조하십시오.
- ONTAP-NAS 및 ONTAP-NAS-Economy 볼륨에 대한 보안 SMB 지원이 추가되었습니다. 이제 Active Directory 사용자 및 그룹을 SMB 볼륨과 함께 사용하여 보안을 강화할 수 있습니다. "보안 SMB 활성화"을(를) 참조하십시오.
- iSCSI 볼륨에 대한 노드 작업의 확장성을 높이기 위해 Trident 노드 동시성이 향상되었습니다.
- LUKS 볼륨을 열 때 공간 재확보를 위한 discard/TRIM 명령을 허용하도록 `--allow-discards` 추가되었습니다.
- LUKS로 암호화된 볼륨을 포맷할 때 성능이 향상되었습니다.
- 실패했지만 부분적으로 포맷된 LUKS 장치에 대한 향상된 LUKS 정리.
- NVMe 볼륨 연결 및 분리를 위한 향상된 Trident 노드 멱등성.
- ONTAP-SAN-Economy 드라이버용 Trident 볼륨 구성에 `internalID` 필드를 추가했습니다.
- NVMe 백엔드용 SnapMirror를 사용한 볼륨 복제 지원이 추가되었습니다. "SnapMirror를 사용하여 볼륨 복제 "를 참조하십시오.

실험적 개선 사항

운영 환경에서는 사용할 수 없습니다.

- [기술 미리보기] `--enable-concurrency` 기능 플래그를 통해 Trident 컨트롤러의 동시 작업을 활성화했습니다. 이를 통해 컨트롤러 작업을 병렬로 실행하여 사용량이 많거나 규모가 큰 환경에서 성능을 향상시킬 수 있습니다.

  이 기능은 실험적인 기능이며 현재 ONTAP-SAN 드라이버(iSCSI 및 FCP 프로토콜)를 사용한 제한된 병렬 워크플로를 지원합니다.

- [Tech Preview] ANF 드라이버에 수동 QOS 지원 기능이 추가되었습니다.

수정 사항

- **Kubernetes**:
  - 기본 SCSI 디스크를 사용할 수 없는 경우 다중 경로 장치의 크기가 일치하지 않을 수 있는 CSI NodeExpandVolume 관련 문제를 수정했습니다.
  - ONTAP-NAS 및 ONTAP-NAS-Economy 드라이버에 대한 중복 엑스포트 정책을 정리하지 못하던 문제를 수정했습니다.
  - `nfsMountOptions`이 설정되지 않은 경우 GCNV 볼륨이 NFSv3로 기본 설정되던 문제를 수정했습니다. 이제 NFSv3 및 NFSv4 프로토콜을 모두 지원합니다. `nfsMountOptions`가 제공되지 않으면 호스트의 기본 NFS 버전(NFSv3 또는 NFSv4)이 사용됩니다.
  - Kustomize를 사용하여 Trident를 설치할 때 발생했던 배포 문제를 수정했습니다("문제 #831").
  - 스냅샷에서 생성된 PVC에 대한 누락된 내보내기 정책을 수정했습니다("문제 #1016").
  - ANF 볼륨 크기가 1GiB 단위로 자동 정렬되지 않던 문제를 수정했습니다.
  - Bottlerocket에서 NFSv3를 사용할 때 발생하던 문제를 수정했습니다.
- 크기 조정 실패에도 불구하고 ONTAP-NAS-Economy 볼륨이 최대 300TB까지 확장되는 문제를 해결했습니다.

- ONTAP REST API를 사용할 때 클론 분할 작업이 동기적으로 수행되던 문제를 해결했습니다.

사용 중단:

- **Kubernetes**: 지원되는 최소 Kubernetes를 v1.27로 업데이트했습니다.

Trident Protect

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너를 기반으로 하는 스테이트풀 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다.

향상된 기능

- 복원 시간이 단축되어 더 자주 전체 백업을 수행할 수 있는 옵션이 제공됩니다.
- GVK(Group-Version-Kind) 필터링을 통해 애플리케이션 정의 및 선택적 복원의 세분성이 향상되었습니다.
- NetApp SnapMirror와 함께 AppMirrorRelationship (AMR)을 사용할 때 전체 PVC 복제를 피하기 위한 효율적인 재동기화 및 역복제.
- EKS Pod Identity를 사용하여 AppVault 버킷을 생성하는 기능이 추가되어 EKS 클러스터의 버킷 자격 증명에 비밀 키를 지정할 필요가 없어졌습니다.
- 필요한 경우 복원 네임스페이스에서 레이블 및 주석 복원을 건너뛸 수 있는 기능이 추가되었습니다.
- AppMirrorRelationship(AMR)은 이제 소스 PVC 확장 여부를 확인하고 필요에 따라 대상 PVC에 적절한 확장을 수행합니다.

수정 사항

- 이전 스냅샷의 스냅샷 주석 값이 최신 스냅샷에 적용되는 버그를 수정했습니다. 이제 모든 스냅샷 주석이 올바르게 적용됩니다.
- 정의되지 않은 경우 기본적으로 데이터 이동 프로그램 암호화(Kopia/Restic)에 대한 시크릿을 정의했습니다.
- S3 appvault 생성에 대한 향상된 유효성 검사 및 오류 메시지가 추가되었습니다.
- AppMirrorRelationship(AMR)은 이제 실패 시도를 방지하기 위해 Bound 상태의 PV만 복제합니다.
- 백업 수가 많은 AppVault에서 AppVaultContent을 가져올 때 오류가 표시되던 문제가 수정되었습니다.
- KubeVirt VMSnapshots는 장애를 방지하기 위해 복원 및 페일오버 작업에서 제외됩니다.
- Kopia에서 Kopia 기본 보존 스케줄이 사용자가 스케줄에서 설정한 내용을 재정의하여 스냅샷이 조기에 제거되는 문제를 수정했습니다.

25.02.1의 변경 사항

Trident

수정 사항

- **Kubernetes**:
  - 기본 이미지 레지스트리가 아닌 다른 이미지 레지스트리를 사용할 때 사이드카 이미지 이름과 버전이 잘못 채워지는 trident-operator의 문제를 수정했습니다("문제 #983").

- ONTAP 페일오버 반환 중에 다중 경로 세션이 복구되지 않는 문제를 수정했습니다("문제 #961").

25.02의 변경 사항

Trident 25.02부터 새로운 기능 요약에서 Trident 및 Trident Protect 릴리스 모두에 대한 개선 사항, 수정 사항 및 사용 중단에 대한 세부 정보를 제공합니다.

Trident

향상된 기능

- **Kubernetes**:
  - iSCSI용 ONTAP ASA r2 지원이 추가되었습니다.
  - 비정상적인 노드 종료 시나리오에서 ONTAP-NAS 볼륨의 강제 분리 지원이 추가되었습니다. 이제 새로운 ONTAP-NAS 볼륨은 Trident에서 관리하는 볼륨별 엑스포트 정책을 활용합니다. 기존 볼륨이 활성 워크로드에 영향을 주지 않고 게시 취소 시 새로운 엑스포트 정책 모델로 전환할 수 있는 업그레이드 경로가 제공됩니다.
  - cloneFromSnapshot 주석을 추가했습니다.
  - 네임스페이스 간 볼륨 클론 지원이 추가되었습니다.
  - iSCSI 자체 복구 스캔 기능을 강화하여 정확한 호스트, 채널, 대상 및 LUN ID를 기준으로 재스캔을 시작할 수 있습니다.
  - Kubernetes 1.32 지원이 추가되었습니다.
- **OpenShift**:
  - ROSA 클러스터에서 RHCOS용 자동 iSCSI 노드 준비 지원이 추가되었습니다.
  - ONTAP 드라이버에 OpenShift Virtualization 지원이 추가되었습니다.
- ONTAP-SAN 드라이버에 Fibre Channel 지원이 추가되었습니다.
- NVMe LUKS 지원이 추가되었습니다.
- 모든 기본 이미지에 대해 스크래치 이미지로 전환했습니다.
- iSCSI 세션이 로그인되어야 하지만 로그인되지 않을 때 iSCSI 연결 상태 검색 및 로깅 기능이 추가되었습니다("문제 #961").
- google-cloud-netapp-volumes 드라이버를 사용하는 SMB 볼륨에 대한 지원이 추가되었습니다.
- ONTAP 볼륨이 삭제 시 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

- **Kubernetes**:
  - 자동 내보내기 정책에서 누락된 노드 IP 주소 문제를 수정했습니다("문제 #965").
  - ONTAP-NAS-Economy에 대해 자동 내보내기 정책이 볼륨별 정책으로 조기에 전환되는 문제를 수정했습니다.
  - 모든 사용 가능한 AWS ARN 파티션을 지원하도록 백엔드 구성 자격 증명을 수정했습니다("문제 #913").
  - Trident 운영자("문제 #924")에서 자동 구성기 조정 기능을 비활성화하는 옵션이 추가되었습니다.

- csi-resizer 컨테이너에 대한 securityContext가 추가되었습니다("문제 #976").

Trident Protect

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너를 기반으로 하는 스테이트풀 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다.

향상된 기능

- KubeVirt / OpenShift Virtualization VM에 대해 volumeMode: File 및 volumeMode: Block(raw device) 스토리지 모두에 대한 백업 및 복원 지원이 추가되었습니다. 이 지원은 모든 Trident 드라이버와 호환되며, Trident Protect와 함께 NetApp SnapMirror를 사용하여 스토리지를 복제할 때 기존 보호 기능을 강화합니다.
- Kubevirt 환경에서 애플리케이션 수준의 동결 동작을 제어할 수 있는 기능이 추가되었습니다.
- AutoSupport 프록시 연결 구성에 대한 지원이 추가되었습니다.
- 데이터 이동 프로그램 암호화(Kopia/Restic)에 사용할 비밀 키를 정의하는 기능을 추가했습니다.
- 실행 후크를 수동으로 실행할 수 있는 기능이 추가되었습니다.
- Trident Protect 설치 중에 보안 컨텍스트 제약 조건(SCC)을 구성할 수 있는 기능을 추가했습니다.
- Trident Protect 설치 중 nodeSelector를 구성하는 기능이 추가되었습니다.
- AppVault 객체에 대한 HTTP/HTTPS 이그레스 프록시 지원이 추가되었습니다.
- 클러스터 범위 리소스를 제외할 수 있도록 ResourceFilter를 확장했습니다.
- S3 AppVault 자격 증명에 AWS 세션 토큰 지원이 추가되었습니다.
- 스냅샷 이전 실행 후크 이후 리소스 수집에 대한 지원이 추가되었습니다.

수정 사항

- ONTAP 볼륨 복구 대기열을 건너뛰도록 임시 볼륨 관리를 개선했습니다.
- SCC 주석이 이제 원래 값으로 복원되었습니다.
- 병렬 작업 지원을 통해 복원 효율성이 향상되었습니다.
- 대규모 애플리케이션에 대한 실행 후크 타임아웃 지원이 강화되었습니다.

24.10.1의 변경 사항

향상된 기능

- **Kubernetes**: Kubernetes 1.32 지원이 추가되었습니다.
- iSCSI 세션이 로그인되어야 하지만 로그인되지 않을 때 iSCSI 연결 상태 검색 및 로깅 기능이 추가되었습니다("문제 #961").

수정 사항

- 자동 내보내기 정책에서 누락된 노드 IP 주소 문제를 수정했습니다("문제 #965").
- ONTAP-NAS-Economy에 대해 자동 내보내기 정책이 볼륨별 정책으로 조기에 전환되는 문제를 수정했습니다.

- CVE-2024-45337 및 CVE-2024-45310을 해결하기 위해 Trident 및 Trident-ASUP 종속성을 업데이트했습니다.
- iSCSI 자체 복구 중에 간헐적으로 비정상적인 비CHAP 포털에 대한 로그아웃이 제거되었습니다("문제 #961").

24.10의 변경 사항

향상된 기능

- Google Cloud NetApp Volumes 드라이버가 이제 NFS 볼륨용으로 정식 출시되었으며 영역 인식 프로비저닝을 지원합니다.
- GCP 워크로드 ID는 GKE를 사용하는 Google Cloud NetApp Volumes의 클라우드 ID로 사용됩니다.
- ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버에 `formatOptions` 구성 매개변수를 추가하여 사용자가 LUN 형식 옵션을 지정할 수 있습니다.
- Azure NetApp Files의 최소 볼륨 크기가 50GiB로 줄어들었습니다. Azure의 새로운 최소 크기는 11월에 정식 출시될 예정입니다.
- `denyNewVolumePools` 구성 매개변수를 추가하여 ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버를 기존 FlexVol 풀로 제한했습니다.
- 모든 ONTAP 드라이버에서 SVM의 애그리게이트 추가, 제거 또는 이름 변경에 대한 감지 기능이 추가되었습니다.
- LUKS LUN에 18MiB 오버헤드를 추가하여 보고된 PVC 크기를 사용할 수 있도록 했습니다.
- ONTAP-SAN 및 ONTAP-SAN-Economy 노드 스테이지 및 언스테이지 오류 처리가 개선되어 스테이지 실패 후 언스테이지에서 장치를 제거할 수 있습니다.
- 고객이 ONTAP에서 Trident에 대한 최소한의 역할을 생성할 수 있도록 사용자 지정 역할 생성기를 추가했습니다.
- 문제 해결을 위해 추가 로깅 기능을 추가했습니다 `lsscsi` ("문제 #792").

Kubernetes

- Kubernetes 네이티브 워크플로우를 위한 새로운 Trident 기능이 추가되었습니다.
  - 데이터 보호
  - 데이터 마이그레이션
  - 재해 복구
  - 애플리케이션 이동성

    "Trident Protect에 대해 자세히 알아보세요".
- Trident가 Kubernetes API 서버와 통신하는 데 사용하는 QPS 값을 설정하는 새로운 플래그 `--k8s-api-qps`를 설치 프로그램에 추가했습니다.
- `--node-prep` 플래그를 설치 프로그램에 추가하여 Kubernetes 클러스터 노드의 스토리지 프로토콜 종속성을 자동으로 관리합니다. Amazon Linux 2023 iSCSI 스토리지 프로토콜과의 호환성을 테스트하고 검증했습니다.
- 비정상적인 노드 종료 시나리오에서 ONTAP-NAS-Economy 볼륨에 대한 강제 분리 지원이 추가되었습니다.
- 새로운 ONTAP-NAS-Economy NFS 볼륨은 `autoExportPolicy` 백엔드 옵션을 사용할 때 qtree별 엑스포트 정책을 사용합니다. Qtree는 액세스 제어 및 보안을 개선하기 위해 게시 시점에만 노드 제한 엑스포트 정책에 매핑됩니다. 기존 qtree는 Trident가 모든 노드에서 볼륨 게시를 취소할 때 활성 워크로드에 영향을 주지 않고 새로운 엑스포트 정책 모델로 전환됩니다.
- Kubernetes 1.31 지원이 추가되었습니다.

실험적 개선 사항

- ONTAP-SAN 드라이버에서 Fibre Channel 지원에 대한 기술 미리 보기가 추가되었습니다.

수정 사항

- **Kubernetes**:
  - Trident Helm 설치를 방지하는 Rancher 승인 웹후크가 수정되었습니다("문제 #839").
  - Helm 차트 값의 고정된 어피니티 키("문제 #898").
  - tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector가 "true" 값("문제 #899")으로 작동하지 않는 문제를 수정했습니다.
  - 클론 생성 중에 생성된 임시 스냅샷이 삭제되었습니다("문제 #901").
- Windows Server 2019 지원이 추가되었습니다.
- Trident 저장소의 `go mod tidy`를 수정했습니다("문제 #767").

사용 중단

- **Kubernetes:**
  - 최소 지원 Kubernetes를 1.25로 업데이트했습니다.
  - POD 보안 정책에 대한 지원이 제거되었습니다.

제품 리브랜딩

24.10 릴리스부터 Astra Trident는 Trident(NetApp Trident)로 브랜드가 변경되었습니다. 이 브랜드 변경은 Trident의 기능, 지원되는 플랫폼 또는 상호 운용성에 영향을 주지 않습니다.

24.06의 변경 사항

향상된 기능

- 중요: `limitVolumeSize` 매개변수는 이제 ONTAP 이코노미 드라이버에서 qtree/LUN 크기를 제한합니다. 해당 드라이버에서 FlexVol 크기를 제어하려면 새 `limitVolumePoolSize` 매개변수를 사용하십시오. ("문제 #341").
- 더 이상 사용되지 않는 igroup이 사용 중인 경우 정확한 LUN ID로 SCSI 스캔을 시작할 수 있는 iSCSI 자체 복구 기능이 추가되었습니다("문제 #883").
- 백엔드가 일시 중단 모드인 경우에도 볼륨 클론 및 크기 조정 작업을 허용하도록 지원 기능을 추가했습니다.
- Trident 컨트롤러에 대한 사용자 구성 로그 설정이 Trident 노드 Pod에 전파될 수 있는 기능이 추가되었습니다.
- ONTAP 버전 9.15.1 이상에서 기본적으로 ONTAPI(ZAPI) 대신 REST를 사용하도록 Trident에 지원 기능을 추가했습니다.
- 새로운 영구 볼륨에 대해 ONTAP 스토리지 백엔드에서 사용자 지정 볼륨 이름 및 메타데이터 지원이 추가되었습니다.
- `azure-netapp-files` (ANF) 드라이버를 개선하여 NFS 마운트 옵션이 NFS 버전 4.x를 사용하도록 설정된 경우 스냅샷 디렉토리가 기본적으로 자동으로 활성화되도록 했습니다.
- NFS 볼륨에 대한 Bottlerocket 지원이 추가되었습니다.
- Google Cloud NetApp Volumes에 대한 기술 미리 보기 지원이 추가되었습니다.

Kubernetes

- Kubernetes 1.30 지원이 추가되었습니다.
- Trident DaemonSet이 시작 시 좀비 마운트와 잔여 추적 파일을 정리하는 기능을 추가했습니다("문제 #883").
- LUKS 볼륨을 동적으로 가져오기 위한 PVC 주석 `trident.netapp.io/luksEncryption`이 추가되었습니다("문제 #849").
- ANF 드라이버에 토폴로지 인식 기능을 추가했습니다.
- Windows Server 2022 노드 지원이 추가되었습니다.

수정 사항

- 오래된 트랜잭션으로 인한 Trident 설치 실패를 수정했습니다.
- Kubernetes에서 오는 경고 메시지를 무시하도록 tridentctl을 수정했습니다("문제 #892").
- Trident 컨트롤러 `SecurityContextConstraint` 우선 순위를 `0` ("문제 #887")로 변경했습니다.
- ONTAP 드라이버는 이제 20MiB 미만의 볼륨 크기를 허용합니다("문제[#885").
- ONTAP-SAN 드라이버의 크기 조정 작업 중 FlexVol 볼륨이 축소되는 것을 방지하기 위해 Trident를 수정했습니다.
- NFS v4.1에서 발생하던 ANF 볼륨 가져오기 오류를 수정했습니다.

24.02의 변경 사항

향상된 기능

- Cloud Identity 지원이 추가되었습니다.
  - ANF를 사용하는 AKS - Azure Workload Identity가 클라우드 ID로 사용됩니다.
  - FSxN을 사용하는 EKS - AWS IAM 역할이 클라우드 ID로 사용됩니다.
- EKS 콘솔에서 EKS 클러스터에 Trident를 애드온으로 설치할 수 있도록 지원 기능을 추가했습니다.
- iSCSI 자체 복구 기능을 구성하고 비활성화할 수 있는 기능이 추가되었습니다("문제 #864").
- AWS IAM 및 SecretsManager와의 통합을 활성화하고 Trident가 백업이 있는 FSx 볼륨을 삭제할 수 있도록 ONTAP 드라이버에 Amazon FSx 특성이 추가되었습니다("문제 #453").

Kubernetes

- Kubernetes 1.29에 대한 지원이 추가되었습니다.

수정 사항

- ACP가 활성화되지 않은 경우 ACP 경고 메시지가 수정되었습니다("문제 #866").
- 클론이 스냅샷과 연결된 경우 ONTAP 드라이버에 대한 스냅샷 삭제 중 클론 분할을 수행하기 전에 10초 지연을 추가했습니다.

사용 중단

- 멀티 플랫폼 이미지 매니페스트에서 in-toto 증명 프레임워크를 제거했습니다.

23.10의 변경 사항

수정 사항

- ontap-nas 및 ontap-nas-flexgroup 스토리지 드라이버의 경우 새롭게 요청된 크기가 전체 볼륨 크기보다 작으면 고정 볼륨 확장이 적용됩니다("문제 #834").
- ontap-nas 및 ontap-nas-flexgroup 스토리지 드라이버의 경우 가져오기 중에 사용 가능한 볼륨 크기만 표시하도록 볼륨 크기를 고정합니다("문제 #722").
- ONTAP-NAS-Economy에 대한 FlexVol 이름 변환 오류를 수정했습니다.
- 노드를 재부팅할 때 Windows 노드에서 Trident 초기화 문제를 수정했습니다.

향상된 기능

Kubernetes

Kubernetes 1.28 지원이 추가되었습니다.

Trident

- azure-netapp-files 스토리지 드라이버와 함께 Azure Managed Identities(AMI)를 사용할 수 있도록 지원이 추가되었습니다.
- ONTAP-SAN 드라이버에 대한 TCP를 통한 NVMe 지원이 추가되었습니다.
- 사용자가 백엔드를 일시 중단 상태로 설정할 때 볼륨 프로비저닝을 일시 중지할 수 있는 기능이 추가되었습니다("문제 #558").

23.07.1의 변경 사항

Kubernetes: 다운타임 없는 업그레이드를 지원하기 위해 데몬셋 삭제 문제를 수정했습니다("문제 #740").

23.07의 변경 사항

수정 사항

Kubernetes

- Trident 업그레이드 시 종료 상태에 멈춰 있는 기존 Pod를 무시하도록 수정했습니다("문제 #740").
- "transient-trident-version-pod" 정의에 toleration이 추가되었습니다("문제 #795").

Trident

- 노드 스테이징 작업 중 고스트 iSCSI 장치를 식별하고 수정하기 위해 LUN 속성을 가져올 때 LUN 일련 번호를 쿼리하도록 ONTAPI(ZAPI) 요청을 수정했습니다.
- 스토리지 드라이버 코드의 오류 처리 문제를 수정했습니다("문제 #816").
- use-rest=true 옵션을 사용하여 ONTAP 드라이버를 사용할 때 할당량 크기 조정 문제가 해결되었습니다.
- ontap-san-economy에서 LUN 클론 생성 오류가 수정되었습니다.
- 게시 정보 필드를 `rawDevicePath`에서 `devicePath`로 되돌립니다. `devicePath` 필드를 채우고 (경우에 따라) 복구하는 로직을 추가했습니다.

향상된 기능

Kubernetes

- 사전 프로비저닝된 스냅샷 가져오기 지원이 추가되었습니다.
- 배포 최소화 및 daemonset Linux 권한("문제 #817").

Trident

- "온라인" 볼륨 및 스냅샷에 대한 상태 필드를 더 이상 보고하지 않습니다.
- ONTAP 백엔드가 오프라인인 경우 백엔드 상태를 업데이트합니다("문제 #801", "#543").
- LUN 일련 번호는 ControllerVolumePublish 워크플로 중에 항상 검색되어 게시됩니다.
- iSCSI 다중 경로 디바이스 일련 번호 및 크기를 확인하기 위한 추가 로직이 추가되었습니다.
- iSCSI 볼륨에 대한 추가 검증을 통해 올바른 다중 경로 디바이스가 준비 해제되었는지 확인합니다.

실험적 개선

ONTAP-SAN 드라이버에 대한 TCP를 통한 NVMe 기술 미리 보기 지원이 추가되었습니다.

설명서

구성 및 형식 면에서 많은 개선이 이루어졌습니다.

사용 중단

Kubernetes

- v1beta1 스냅샷 지원이 제거되었습니다.
- CSI 이전 볼륨 및 스토리지 클래스에 대한 지원이 제거되었습니다.
- 최소 지원 Kubernetes를 1.22로 업데이트했습니다.

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 `--enable-force-detach` Trident 설치 프로그램 플래그를 사용하여 활성화해야 합니다.

수정 사항

- 사양에 지정된 경우 설치 시 IPv6 localhost를 사용하도록 Trident Operator를 수정했습니다.
- Trident Operator 클러스터 역할 권한이 번들 권한과 동기화되도록 수정했습니다("문제 #799").
- RWX 모드에서 여러 노드에 원시 블록 볼륨을 연결할 때 발생하던 문제를 수정했습니다.
- FlexGroup 복제 지원 및 SMB 볼륨에 대한 볼륨 가져오기가 수정되었습니다.
- Trident 컨트롤러가 즉시 종료되지 않던 문제를 수정했습니다("문제 #811").
- ontap-san-* 드라이버로 프로비저닝된 지정된 LUN과 연결된 모든 igroup 이름을 나열하도록 수정 사항이 추가되었습니다.

- 외부 프로세스가 완료될 때까지 실행될 수 있도록 수정 사항을 추가했습니다.
- s390 아키텍처에 대한 컴파일 오류를 수정했습니다("문제 #537").
- 볼륨 마운트 작업 중 잘못된 로깅 레벨이 수정되었습니다("문제 #781").
- 잠재적인 유형 어설션 오류를 수정했습니다("문제 #802").

향상된 기능

- Kubernetes:
  - Kubernetes 1.27에 대한 지원이 추가되었습니다.
  - LUKS 볼륨 가져오기 지원이 추가되었습니다.
  - ReadWriteOncePod PVC 액세스 모드 지원이 추가되었습니다.
  - 비정상적인 노드 종료 시나리오에서 ONTAP-SAN-* 볼륨에 대한 강제 분리 지원이 추가되었습니다.
  - 이제 모든 ONTAP-SAN-* 볼륨은 노드별 igroup을 사용합니다. LUN은 보안 강화를 위해 해당 노드에 활발하게 게시되는 동안에만 igroup에 매핑됩니다. 기존 볼륨은 Trident가 활성 워크로드에 영향을 주지 않고 안전하다고 판단하는 시점에 새로운 igroup 체계로 전환됩니다("문제 #758").
  - ONTAP-SAN-* 백엔드에서 사용되지 않는 Trident 관리 igroup을 정리하여 Trident 보안을 개선했습니다.
- ontap-nas-economy 및 ontap-nas-flexgroup 스토리지 드라이버에 Amazon FSx를 사용하는 SMB 볼륨 지원이 추가되었습니다.
- ontap-nas, ontap-nas-economy 및 ontap-nas-flexgroup 스토리지 드라이버에서 SMB 공유에 대한 지원이 추가되었습니다.
- arm64 노드 지원이 추가되었습니다("문제 #732").
- API 서버를 먼저 비활성화하여 Trident 종료 절차를 개선했습니다("문제 #811").
- Makefile에 Windows 및 arm64 호스트용 크로스 플랫폼 빌드 지원을 추가했습니다. BUILD.md를 참조하십시오.

사용 중단

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

23.01.1의 변경 사항

수정 사항

- 사양에 지정된 경우 설치 시 IPv6 localhost를 사용하도록 Trident Operator를 수정했습니다.
- Trident Operator 클러스터 역할 권한이 번들 권한과 동기화되도록 수정했습니다"문제 #799".
- 외부 프로세스가 완료될 때까지 실행될 수 있도록 수정 사항을 추가했습니다.
- RWX 모드에서 여러 노드에 원시 블록 볼륨을 연결할 때 발생하던 문제를 수정했습니다.
- FlexGroup 복제 지원 및 SMB 볼륨에 대한 볼륨 가져오기가 수정되었습니다.

23.01의 변경 사항

Kubernetes 1.27은 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 업그레이드하십시오.

수정 사항

- Kubernetes: Helm을 통한 Trident 설치 문제를 해결하기 위해 Pod Security Policy 생성을 제외하는 옵션을 추가했습니다("이슈 #783, #794").

향상된 기능

Kubernetes

- Kubernetes 1.26 지원이 추가되었습니다.
- 전반적인 Trident RBAC 리소스 활용률 개선("문제 #757").
- 호스트 노드에서 손상되었거나 만료된 iSCSI 세션을 감지하고 수정하는 자동화 기능을 추가했습니다.
- LUKS 암호화된 볼륨 확장에 대한 지원이 추가되었습니다.
- Kubernetes: LUKS 암호화 볼륨에 대한 자격 증명 교체 지원이 추가되었습니다.

Trident

- ontap-nas 스토리지 드라이버에 Amazon FSx for NetApp ONTAP를 사용하는 SMB 볼륨 지원이 추가되었습니다.
- SMB 볼륨 사용 시 NTFS 권한 지원이 추가되었습니다.
- CVS 서비스 수준을 사용하는 GCP 볼륨의 스토리지 풀에 대한 지원이 추가되었습니다.
- ontap-nas-flexgroup 스토리지 드라이버로 FlexGroups를 생성할 때 flexgroupAggregateList의 선택적 사용을 지원합니다.
- 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 Kubernetes를 1.21로 업데이트했습니다.
- `ontap-san` 또는 `ontap-san-economy` 드라이버를 구성할 때 DataLIF를 더 이상 지정하지 않아야 합니다.

22.10의 변경 사항

Trident 22.10으로 업그레이드하기 전에 다음의 중요 정보를 반드시 읽어주십시오**.**

Trident 22.10에 대한 중요 정보

- Kubernetes 1.25는 이제 Trident에서 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Trident를 22.10으로 업그레이드해야 합니다.
- Trident는 이제 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

수정 사항

- `credentials` 필드를 사용하여 생성된 ONTAP 백엔드가 22.07.0 업그레이드 중에 온라인 상태가 되지 않는 특정 문제를 수정했습니다("문제 #759").
- **Docker:** 일부 환경("문제 #548" 및 "문제 #760")에서 Docker 볼륨 플러그인이 시작되지 않는 문제를 수정했습니다.
- ONTAP SAN 백엔드와 관련된 SLM 문제를 수정하여 보고 노드에 속하는 dataLIF의 하위 집합만 게시되도록 했습니다.
- 볼륨 연결 시 iSCSI LUN에 대한 불필요한 스캔이 발생하던 성능 문제를 해결했습니다.
- Trident iSCSI 워크플로우 내에서 세부적인 재시도를 제거하여 빠른 실패를 유도하고 외부 재시도 간격을 줄였습니다.
- 해당 다중 경로 장치가 이미 플러시된 경우 iSCSI 장치를 플러시할 때 오류가 반환되던 문제를 수정했습니다.

향상된 기능

- Kubernetes:
  - Kubernetes 1.25 지원이 추가되었습니다. Kubernetes 1.25로 업그레이드하기 전에 Trident를 22.10으로 업그레이드해야 합니다.
  - 향후 권한 향상을 위해 Trident Deployment와 DaemonSet에 대해 별도의 ServiceAccount, ClusterRole, ClusterRoleBinding이 추가되었습니다.
  - "네임스페이스 간 볼륨 공유"에 대한 지원이 추가되었습니다.
- 이제 모든 Trident `ontap-*` 스토리지 드라이버는 ONTAP REST API와 호환됩니다.
- 새로운 operator yaml (`bundle_post_1_25.yaml`)을(를) 추가하여 `PodSecurityPolicy` 없이 Kubernetes 1.25를 지원합니다.
- "LUKS로 암호화된 볼륨 지원"가 `ontap-san` 및 `ontap-san-economy` 스토리지 드라이버에 추가되었습니다.
- Windows Server 2019 노드 지원이 추가되었습니다.
- "Windows 노드에서 SMB 볼륨 지원"를 `azure-netapp-files` 스토리지 드라이버를 통해 추가했습니다.
- ONTAP 드라이버용 자동 MetroCluster 전환 감지가 이제 일반적으로 제공됩니다.

사용 중단

- **Kubernetes:** 지원되는 최소 Kubernetes 버전을 1.20으로 업데이트했습니다.

- Astra Data Store(ADS) 드라이버를 제거했습니다.
- iSCSI를 위한 워커 노드 다중 경로를 구성할 때 `find_multipaths`에 대한 `yes` 및 `smart` 옵션 지원이 제거되었습니다.

22.07의 변경 사항

수정 사항

Kubernetes

- Helm 또는 Trident Operator를 사용하여 Trident를 구성할 때 노드 선택기의 부울 및 숫자 값을 처리하는 문제를 수정했습니다. ("GitHub 이슈 #700")
- CHAP 경로가 아닌 경로에서 발생하는 오류 처리 문제를 수정하여 kubelet이 실패할 경우 재시도하도록 했습니다. "GitHub 이슈 #736")

향상된 기능

- CSI 이미지의 기본 레지스트리를 k8s.gcr.io에서 registry.k8s.io로 전환
- ONTAP-SAN 볼륨은 이제 노드별 igroup을 사용하며 보안 태세를 개선하기 위해 해당 노드에 활성 게시되는 동안에만 LUN을 igroup에 매핑합니다. 기존 볼륨은 Trident가 활성 워크로드에 영향을 주지 않고 안전하다고 판단할 때 새로운 igroup 체계로 전환됩니다.
- Trident 설치 시 ResourceQuota를 포함하여 기본적으로 PriorityClass 사용이 제한될 때 Trident DaemonSet이 스케줄되도록 했습니다.
- Azure NetApp Files 드라이버에 네트워크 기능 지원이 추가되었습니다. ("GitHub 이슈 #717")
- ONTAP 드라이버에 기술 미리보기 자동 MetroCluster 전환 감지 기능이 추가되었습니다. ("GitHub 이슈 #228")

사용 중단

- **Kubernetes:** 지원되는 최소 Kubernetes를 1.19로 업데이트했습니다.
- 백엔드 구성에서 더 이상 단일 구성에 여러 인증 유형을 허용하지 않습니다.

제거

- AWS CVS 드라이버(22.04부터 더 이상 사용되지 않음)가 제거되었습니다.
- Kubernetes
  - 노드 Pod에서 불필요한 SYS_ADMIN 기능을 제거했습니다.
  - 노드 준비 과정을 간단한 호스트 정보 및 활성 서비스 검색으로 축소하여 워커 노드에서 NFS/iSCSI 서비스를 사용할 수 있는지 최선을 다해 확인합니다.

설명서

Trident 설치 시 활성화되는 권한에 대한 자세한 내용을 담은 새로운 "Pod 보안 표준" (PSS) 섹션이 추가되었습니다.

22.04의 변경 사항

NetApp은 제품과 서비스를 지속적으로 개선하고 향상시키고 있습니다. 다음은 Trident의 최신 기능 중 일부입니다.

이전 릴리스에 대한 정보는 "이전 버전의 문서"를 참조하십시오.

> 이전 Trident 릴리스에서 업그레이드하고 Azure NetApp Files를 사용하는 경우 `location` 구성 매개변수는 이제 필수 단일 필드입니다.

수정 사항

- iSCSI 이니시에이터 이름 구문 분석 기능이 개선되었습니다. ("GitHub 이슈 #681")
- CSI 스토리지 클래스 매개변수가 허용되지 않던 문제를 수정했습니다. ("GitHub 이슈 #598")
- Trident CRD에서 중복된 키 선언을 수정했습니다. ("GitHub 이슈 #671")
- 부정확했던 CSI 스냅샷 로그를 수정했습니다. ("GitHub 이슈 #629"))
- 삭제된 노드에서 볼륨 게시 취소 관련 문제를 수정했습니다. ("GitHub 이슈 #691")
- 블록 장치에서 파일 시스템 불일치 처리 기능이 추가되었습니다. ("GitHub 이슈 #656")
- 설치 중 `imageRegistry` 플래그를 설정할 때 자동 지원 이미지를 가져오는 문제를 수정했습니다. ("GitHub 이슈 #715")
- Azure NetApp Files 드라이버가 여러 내보내기 규칙이 있는 볼륨을 복제하지 못하던 문제를 수정했습니다.

향상된 기능

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

제거

E-Series 드라이버(20.07 이후 비활성화됨)가 제거되었습니다.

22.01.1의 변경 사항

수정 사항

- 삭제된 노드에서 볼륨 게시 취소 관련 문제를 수정했습니다. ("GitHub 이슈 #691")
- ONTAP API 응답에서 집계 공간에 대한 nil 필드에 액세스할 때 발생하던 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 있었던 문제를 수정했습니다.
- ONTAP SAN IPv6 DataLIF는 이제 대괄호로 지정하면 작동합니다.
- 이미 가져온 볼륨을 다시 가져오려고 하면 EOF가 반환되어 PVC가 보류 상태로 남아 있던 문제를 수정했습니다.

("GitHub 이슈 #489")

- SolidFire 볼륨에 스냅샷이 32개 이상 생성될 때 Trident 성능이 저하되는 문제를 수정했습니다.
- SSL 인증서 생성 시 SHA-1을 SHA-256으로 대체했습니다.
- Azure NetApp Files 드라이버에서 리소스 이름 중복 허용 및 작업 제한을 단일 위치로 설정하는 수정 사항이 적용되었습니다.
- Azure NetApp Files 드라이버에서 리소스 이름 중복 허용 및 작업 제한을 단일 위치로 설정하는 수정 사항이 적용되었습니다.

향상된 기능

- Kubernetes 개선 사항:
  - Kubernetes 1.23에 대한 지원이 추가되었습니다.
  - Trident Operator 또는 Helm을 통해 설치될 때 Trident Pod에 대한 스케줄링 옵션을 추가합니다. ("GitHub 이슈 #651")
- GCP 드라이버에서 지역 간 볼륨을 허용합니다. ("GitHub 이슈 #633")
- Azure NetApp Files 볼륨에 'unixPermissions' 옵션 지원이 추가되었습니다. ("GitHub 이슈 #666")

사용 중단

Trident REST 인터페이스는 127.0.0.1 또는 [::1] 주소에서만 수신 및 서비스를 제공할 수 있습니다

21.10.1의 변경 사항

> v21.10.0 릴리스에는 노드를 Kubernetes 클러스터에서 제거한 후 다시 추가할 때 Trident 컨트롤러가 CrashLoopBackOff 상태에 빠질 수 있는 문제가 있습니다. 이 문제는 v21.10.1에서 수정되었습니다(GitHub 이슈 669).

수정 사항

- GCP CVS 백엔드에서 볼륨을 가져올 때 가져오기 실패를 초래하는 잠재적 경합 상태를 수정했습니다.
- 노드를 제거했다가 Kubernetes 클러스터에 다시 추가할 때 Trident 컨트롤러가 CrashLoopBackOff 상태에 빠질 수 있는 문제를 수정했습니다(GitHub 이슈 669).
- SVM 이름이 지정되지 않은 경우 SVM이 더 이상 검색되지 않던 문제를 수정했습니다(GitHub 이슈 612).

21.10.0의 변경 사항

수정 사항

- XFS 볼륨의 복제본이 원본 볼륨과 동일한 노드에 마운트될 수 없었던 문제를 수정했습니다(GitHub 이슈 514).
- Trident 종료 시 치명적인 오류를 기록하는 문제를 수정했습니다(GitHub 이슈 597).
- Kubernetes 관련 수정 사항:
  - `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버를 사용하여 스냅샷을 생성할 때 볼륨의 사용 공간을 최소 restoreSize로 반환합니다(GitHub 이슈 645).

- 볼륨 크기 조정 후 `Failed to expand filesystem` 오류가 기록되던 문제를 수정했습니다(GitHub 이슈 560).
- Pod가 `Terminating` 상태에서 멈추는 문제를 수정했습니다(GitHub 이슈 572).
- `ontap-san-economy` FlexVol에 스냅샷 LUN이 가득 차는 문제를 수정했습니다(GitHub 이슈 533).
- 다른 이미지와 관련된 사용자 지정 YAML 설치 프로그램 문제를 수정했습니다(GitHub 이슈 613).
- 스냅샷 크기 계산 오류를 수정했습니다(GitHub 이슈 611).
- 모든 Trident 설치 프로그램이 일반 Kubernetes를 OpenShift로 식별할 수 있는 문제를 수정했습니다(GitHub 이슈 639).
- Kubernetes API 서버에 연결할 수 없는 경우 Trident 운영자가 조정을 중지하도록 수정했습니다(GitHub 문제 599).

향상된 기능

- GCP-CVS Performance 볼륨에 대한 `unixPermissions` 옵션 지원이 추가되었습니다.
- GCP에서 600 GiB~1 TiB 범위의 스케일 최적화 CVS 볼륨에 대한 지원이 추가되었습니다.
- Kubernetes 관련 개선 사항:
  - Kubernetes 1.22에 대한 지원이 추가되었습니다.
  - Kubernetes 1.22에서 Trident 오퍼레이터와 Helm 차트가 작동하도록 설정했습니다(GitHub 이슈 628).
  - `tridentctl` images 명령에 operator 이미지를 추가했습니다(GitHub 이슈 570).

실험적 개선 사항

- `ontap-san` 드라이버에 볼륨 복제 지원 기능이 추가되었습니다.
- `ontap-nas-flexgroup`, `ontap-san` 및 `ontap-nas-economy` 드라이버에 대한 기술 미리보기 REST 지원이 추가되었습니다.

알려진 문제점

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

- Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상 버전으로 업그레이드할 때는 클러스터를 업그레이드하기 전에 values.yaml을 업데이트하여 `excludePodSecurityPolicy`을 `true`로 설정하거나 `--set excludePodSecurityPolicy=true`을 `helm upgrade` 명령에 추가해야 합니다.
- Trident는 이제 StorageClass에 `fsType`가 지정되지 않은 볼륨에 대해 빈 `fsType` (`fsType=""`를 강제 적용합니다. `Kubernetes 1.17` 이상에서 작업할 때, `Trident`는 `NFS` 볼륨에 대해 빈 `fsType` 제공을 지원합니다. iSCSI 볼륨의 경우, Security Context를 사용하여 `fsGroup`를 강제 적용할 때 StorageClass에 `fsType`를 설정해야 합니다.
- 여러 Trident 인스턴스에서 백엔드를 사용하는 경우 각 백엔드 구성 파일은 ONTAP 백엔드에 대해 다른 `storagePrefix` 값을 가지거나 SolidFire 백엔드에 대해 다른 `TenantName`를 사용해야 합니다. `Trident`는 다른 `Trident` 인스턴스가 생성한 볼륨을 감지할 수 없습니다. `ONTAP` 또는 `SolidFire` 백엔드에서 기존 볼륨을 생성하려고 하면 성공합니다. `Trident`가 볼륨 생성을 멱등성 작업으로 처리하기 때문입니다. `storagePrefix` 또는 `TenantName`가 다르지 않으면 동일한 백엔드에서 생성된 볼륨에 대해 이름 충돌이 발생할 수 있습니다.

- Trident를 설치할 때(`tridentctl` 또는 Trident Operator 사용) `tridentctl`를 사용하여 `Trident`를 관리하는 경우 `KUBECONFIG` 환경 변수가 설정되어 있는지 확인해야 합니다. 이는 `tridentctl`가 작업할 `Kubernetes` 클러스터를 지정하는 데 필요합니다. 여러 `Kubernetes` 환경을 사용하는 경우 `KUBECONFIG` 파일이 정확하게 소싱되었는지 확인해야 합니다.
- iSCSI PV에 대한 온라인 공간 재확보를 수행하려면 워커 노드의 기본 OS에서 볼륨에 마운트 옵션을 전달해야 할 수 있습니다. 이는 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 인스턴스에 해당하며 `discard` "마운트 옵션"온라인 블록 폐기를 지원하려면[`StorageClass`에 discard mountOption이 포함되어 있는지 확인하십시오.
- Kubernetes 클러스터당 Trident 인스턴스가 두 개 이상 있는 경우 Trident는 다른 인스턴스와 통신할 수 없으며 다른 인스턴스가 생성한 다른 볼륨을 검색할 수 없습니다. 따라서 클러스터 내에서 두 개 이상의 인스턴스가 실행될 경우 예기치 않은 잘못된 동작이 발생합니다. Kubernetes 클러스터당 Trident 인스턴스는 하나만 있어야 합니다.
- Trident 기반 `StorageClass` 객체가 Trident 오프라인 상태일 때 Kubernetes에서 삭제되면 Trident가 다시 온라인 상태가 되더라도 데이터베이스에서 해당 스토리지 클래스를 제거하지 않습니다. `tridentctl` 또는 REST API를 사용하여 이러한 스토리지 클래스를 삭제해야 합니다.
- 사용자가 해당 PVC를 삭제하기 전에 Trident에서 프로비저닝한 PV를 삭제하는 경우 Trident는 백업 볼륨을 자동으로 삭제하지 않습니다. `tridentctl` 또는 REST API를 통해 볼륨을 제거해야 합니다.
- ONTAP는 각 프로비저닝 요청에 대해 애그리게이트 세트가 고유하지 않는 한 한 번에 둘 이상의 FlexGroup을 동시에 프로비저닝할 수 없습니다.
- IPv6를 통해 Trident를 사용할 때는 백엔드 정의에서 대괄호 안에 `managementLIF` 및 `dataLIF`를 지정해야 합니다. 예를 들어, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`입니다.

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

- `solidfire-san` 드라이버를 OpenShift 4.5에서 사용하는 경우 기본 워커 노드가 MD5를 CHAP 인증 알고리즘으로 사용하는지 확인하십시오. Element 12.7에서는 안전한 FIPS 규격 준수 CHAP 알고리즘인 SHA1, SHA-256 및 SHA3-256을 사용할 수 있습니다.

자세한 정보 찾기

- "Trident GitHub"
- "Trident 블로그"

이전 버전의 문서

Trident 25.10을 실행 중이 아닌 경우 이전 릴리스에 대한 문서는 "Trident 지원 라이프사이클"을 기반으로 제공됩니다.

- "Trident 25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"

- "Trident 23.04"
- "Trident 23.01"

NetApp Trident ONTAP ASA r2 스토리지 시스템 지원

NetApp Trident 25.02 이상 버전은 NetApp ASA r2 시스템을 스토리지 백엔드로 지원합니다. 자세한 내용은 "ASA r2 시스템"를 참조하십시오.

ASA r2 시스템에는 `ontap-san` 드라이버가 필요합니다. Trident는 ASA r2 시스템용 `ontap-san-economy` 드라이버를 지원하지 않습니다.

백엔드 구성에서 `ontap-san`를 `storageDriverName`로 지정하면, Trident가 ASA r2 스토리지 시스템을 자동으로 감지합니다.

Trident는 Trident protect를 통해 ASA r2 시스템에 제한적인 데이터 보호 기능을 제공합니다.

지원되는 SAN 프로토콜은 Trident 버전에 따라 다릅니다.

- Trident 25.02 이상에서는 iSCSI를 지원합니다.
- Trident 25.06 이상에서는 iSCSI 외에도 NVMe/TCP를 지원합니다.

ONTAP 백엔드 스토리지의 경우 스토리지 가상 머신(SVM)에 하나 이상의 애그리게이트를 할당해야 합니다. 지침은 "ASA r2 시스템에서 SVM에 애그리게이트 할당"을(를) 참조하십시오.

지원되는 작업

- 영구 볼륨(PV) 프로비저닝
- 동적 볼륨 프로비저닝
- 볼륨 생성 및 삭제
- 볼륨 클론 생성
- 볼륨 확장
- 스토리지 클래스 관리

지원되지 않는 작업

- LUKS 암호화
- SnapMirror 볼륨 복제
- 총 사용량 제한
- 공간 예약 모드
- 스냅샷
- 계층화

자세한 내용은 "ONTAP SAN 구성 옵션 및 예"을 참조하십시오.

알려진 문제점

알려진 문제는 이 제품 릴리스를 성공적으로 사용하지 못하게 할 수 있는 문제를 식별합니다.

현재 릴리스에 영향을 미치는 알려진 문제점은 다음과 같습니다.

대용량 파일의 **Restic** 백업 복원이 실패할 수 있습니다

Restic을 사용하여 생성된 Amazon S3 백업에서 30GB 이상의 파일을 복원할 때 복원 작업이 실패할 수 있습니다. 해결 방법으로 Kopia를 데이터 이동 도구로 사용하여 데이터를 백업하십시오(Kopia는 백업 시 기본 데이터 이동 도구입니다). 지침은 "Trident Protect를 사용하여 애플리케이션을 보호하세요."를 참조하십시오.

시작하기

Trident에 대해 알아보십시오

Trident에 대해 알아보십시오

Trident는 NetApp에서 관리하는 완전히 지원되는 오픈 소스 프로젝트입니다. Container Storage Interface(CSI)와 같은 업계 표준 인터페이스를 사용하여 컨테이너화된 애플리케이션의 영구 저장 요구 사항을 충족하도록 설계되었습니다.

Trident란 무엇입니까?

NetApp Trident를 사용하면 퍼블릭 클라우드 또는 온프레미스의 모든 주요 NetApp 스토리지 플랫폼에서 스토리지 리소스를 사용하고 관리할 수 있습니다. 여기에는 온프레미스 ONTAP 클러스터(AFF, FAS 및 ASA), ONTAP Select, Cloud Volumes ONTAP, Element 소프트웨어(NetApp HCI, SolidFire), Azure NetApp Files 및 Amazon FSx for NetApp ONTAP가 포함됩니다.

Trident는 Container Storage Interface(CSI)를 준수하는 동적 스토리지 오케스트레이터로, "Kubernetes"와 기본적으로 통합됩니다. Trident는 클러스터의 각 워커 노드에서 단일 Controller Pod와 Node Pod로 실행됩니다. 자세한 내용은 "Trident 아키텍처"를 참조하십시오.

Trident는 NetApp 스토리지 플랫폼을 위한 Docker 에코시스템과의 직접적인 통합도 제공합니다. NetApp Docker 볼륨 플러그인(nDVP)은 스토리지 플랫폼에서 Docker 호스트로 스토리지 리소스를 프로비저닝하고 관리하는 기능을 지원합니다. 자세한 내용은 "Docker용 Trident 배포"를 참조하십시오.

Kubernetes를 처음 사용하는 경우 "Kubernetes 개념 및 도구"에 익숙해져야 합니다.

NetApp 제품과 Kubernetes 통합

NetApp의 스토리지 제품 포트폴리오는 Kubernetes 클러스터의 다양한 측면과 통합되어 고급 데이터 관리 기능을 제공함으로써 Kubernetes 배포의 기능, 성능, 가용성을 향상시킵니다.

Amazon FSx for NetApp ONTAP

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

Azure NetApp Files

"Azure NetApp Files"는 NetApp이 제공하는 엔터프라이즈급 Azure 파일 공유 서비스입니다. Azure에서 기본적으로 가장 까다로운 파일 기반 워크로드를 실행할 수 있으며, NetApp에서 기대할 수 있는 성능과 풍부한 데이터 관리 기능을 제공합니다.

Cloud Volumes ONTAP

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

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes"는 Google Cloud에서 제공되는 완전 관리형 파일 스토리지 서비스로, 고성능의 엔터프라이즈급 파일 스토리지를 제공합니다.

Element 소프트웨어

"요소"를 사용하면 스토리지 관리자가 성능을 보장하고 간소화되고 효율적인 스토리지 설치 공간을 구현하여 워크로드를 통합할 수 있습니다.

NetApp HCI

"NetApp HCI" 일상적인 작업을 자동화하고 인프라 관리자가 더 중요한 기능에 집중할 수 있도록 함으로써 데이터 센터의 관리 및 확장을 간소화합니다.

Trident는 기본 NetApp HCI 스토리지 플랫폼에 대해 직접 컨테이너화된 애플리케이션용 스토리지 디바이스를 프로비저닝하고 관리할 수 있습니다.

NetApp ONTAP

"NetApp ONTAP"는 모든 애플리케이션에 고급 데이터 관리 기능을 제공하는 NetApp 멀티프로토콜 통합 스토리지 운영 체제입니다.

ONTAP 시스템은 올플래시, 하이브리드 또는 올HDD 구성을 제공하며 온프레미스 FAS, AFA 및 ASA 클러스터, ONTAP Select 및 Cloud Volumes ONTAP 등 다양한 구축 모델을 제공합니다. Trident는 이러한 ONTAP 구축 모델을 지원합니다.

Trident 아키텍처

Trident는 클러스터의 각 워커 노드에서 단일 Controller Pod와 Node Pod로 실행됩니다. Node Pod는 Trident 볼륨을 마운트하려는 모든 호스트에서 실행되어야 합니다.

컨트롤러 **Pod** 및 노드 **Pod** 이해

Trident는 Kubernetes 클러스터에 단일 Trident 컨트롤러 Pod 및 하나 이상의 Trident 노드 Pod로 배포되며 표준 Kubernetes _CSI Sidecar Containers_를 사용하여 CSI 플러그인 배포를 간소화합니다. "Kubernetes CSI 사이드카 컨테이너"는 Kubernetes Storage 커뮤니티에서 유지 관리합니다.

Kubernetes "노드 선택기" 및 "톨러레이션 및 테인트"는 파드가 특정 또는 선호하는 노드에서 실행되도록 제한하는 데 사용됩니다. Trident 설치 중에 컨트롤러 및 노드 파드에 대한 노드 선택기 및 허용 조건을 구성할 수 있습니다.

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

- 노드 플러그인은 스토리지를 노드에 연결하는 작업을 처리합니다.

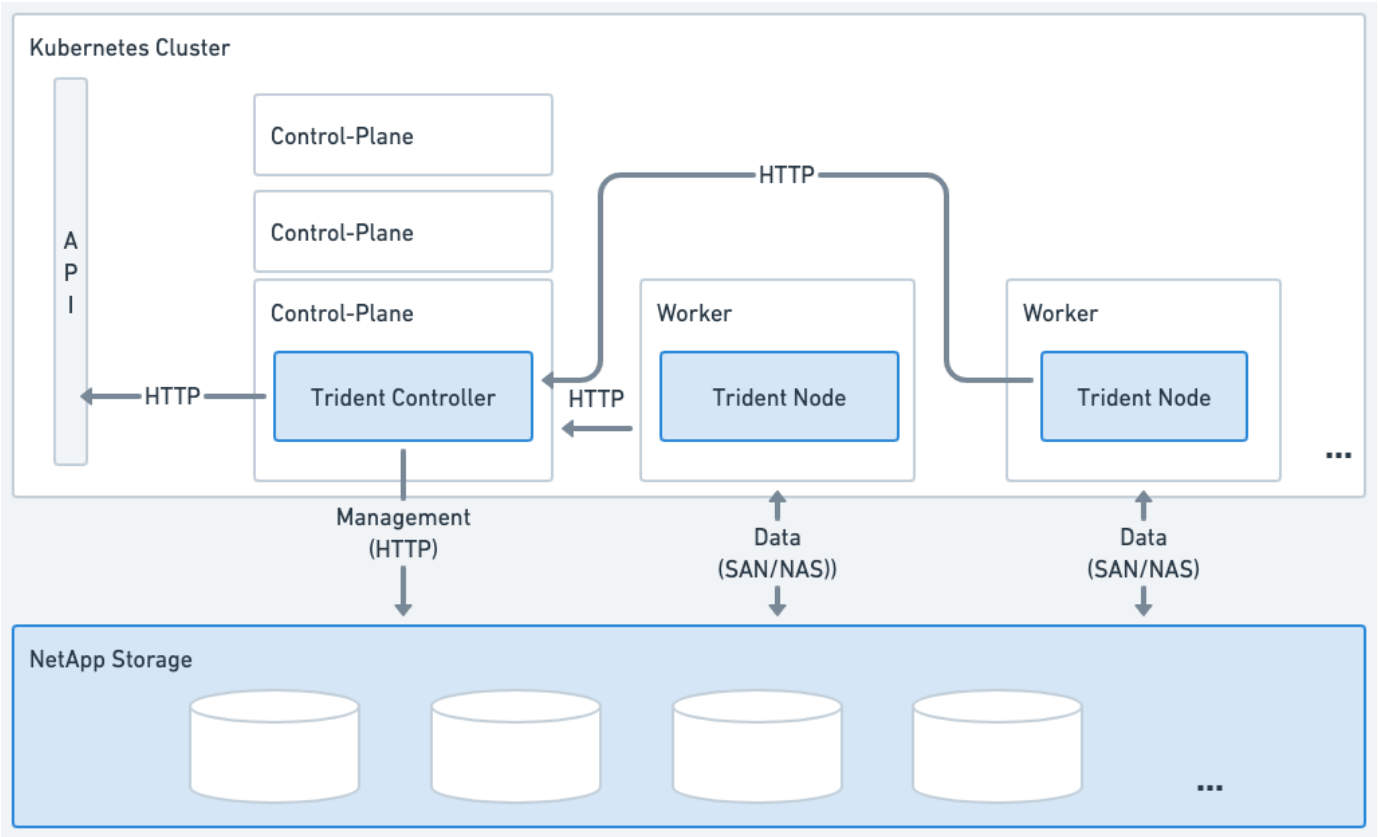


그림 **1. Kubernetes** 클러스터에 배포된 **Trident**

Trident 컨트롤러 **Pod**

Trident Controller Pod는 CSI Controller 플러그인을 실행하는 단일 Pod입니다.

- NetApp 스토리지에서 볼륨 프로비저닝 및 관리를 담당합니다.
- Kubernetes 배포에 의해 관리됨
- 설치 매개변수에 따라 컨트롤 플레인 또는 워커 노드에서 실행할 수 있습니다.

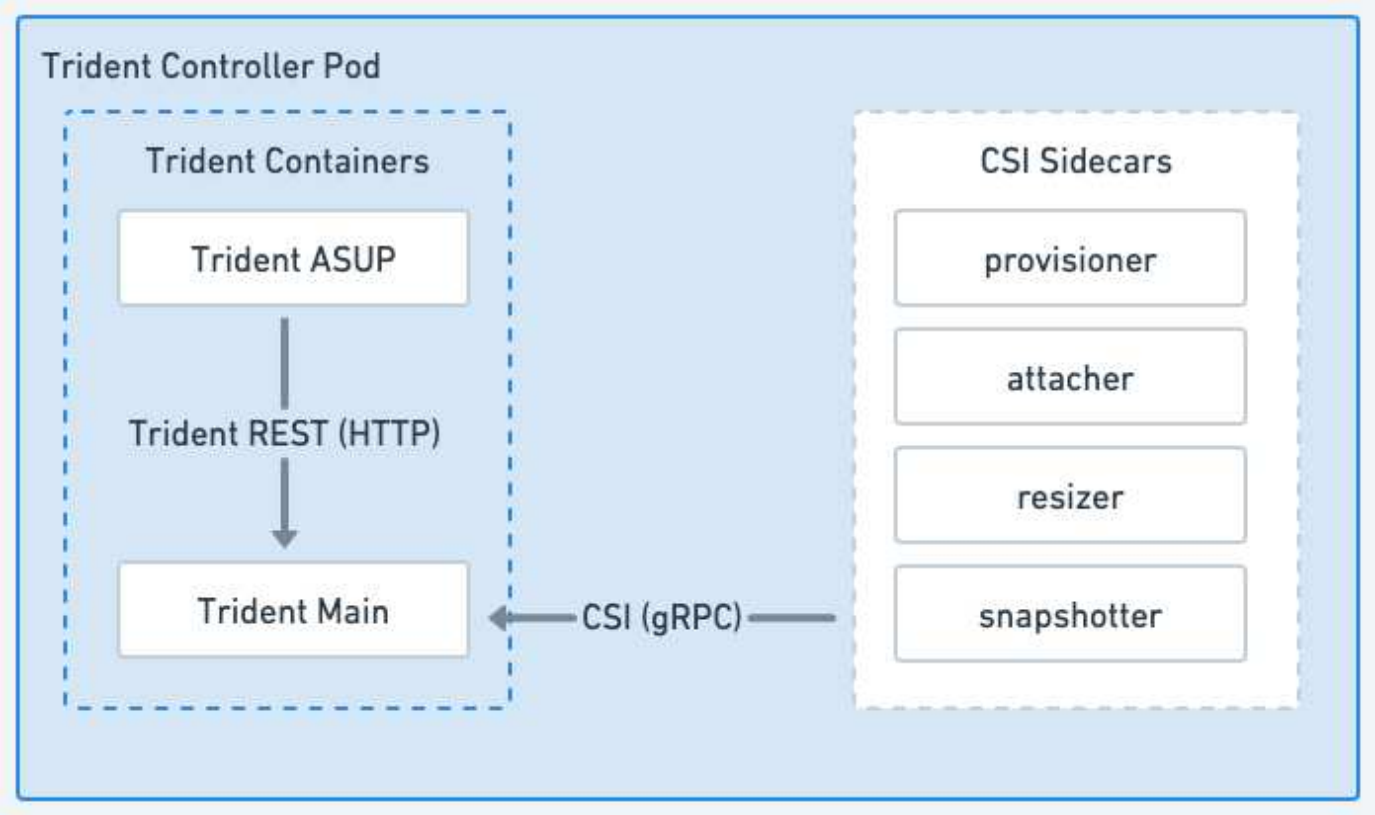


그림 **2. Trident** 컨트롤러 **Pod** 다이어그램

Trident 노드 **Pod**

Trident Node Pod는 CSI Node 플러그인을 실행하는 특권 Pod입니다.

- 호스트에서 실행 중인 Pod에 대한 스토리지 마운트 및 마운트 해제를 담당합니다
- Kubernetes DaemonSet에서 관리
- NetApp 스토리지를 마운트할 모든 노드에서 실행되어야 합니다

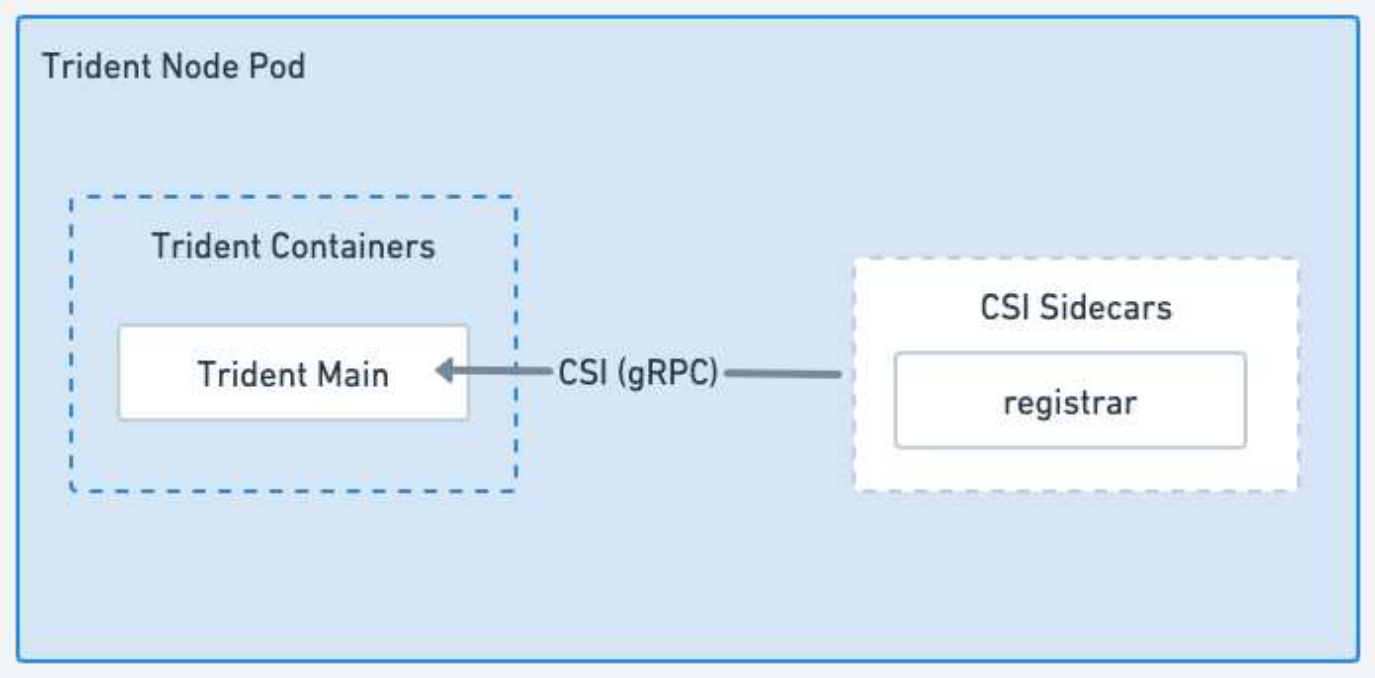


그림 **3. Trident** 노드 **Pod** 다이어그램

지원되는 **Kubernetes** 클러스터 아키텍처

Trident는 다음과 같은 Kubernetes 아키텍처를 지원합니다.

Kubernetes 클러스터 아키텍처	지원됨	기본 설치
단일 마스터, 컴퓨팅	예	예
다중 마스터, 컴퓨팅	예	예
마스터, `etcd` 컴퓨팅	예	예
마스터, 인프라, 컴퓨팅	예	예

개념

프로비저닝

Trident에서 프로비저닝은 크게 두 단계로 구성됩니다. 첫 번째 단계는 스토리지 클래스를 적합한 백엔드 스토리지 풀 세트와 연결하는 단계로, 프로비저닝 전에 필요한 준비 작업입니다. 두 번째 단계는 볼륨 생성 자체를 포함하며, 생성 예정인 볼륨의 스토리지 클래스와 연결된 스토리지 풀 중에서 하나를 선택해야 합니다.

스토리지 클래스 연결

백엔드 스토리지 풀을 스토리지 클래스와 연결하는 작업은 스토리지 클래스에서 요청한 속성과 해당 클래스의 `storagePools`, `additionalStoragePools` 및 `excludeStoragePools` 목록을 모두 활용합니다. 스토리지 클래스를 생성할 때 Trident는 각 백엔드가 제공하는 속성과 풀을 스토리지 클래스에서 요청한 속성 및 풀과 비교합니다.

스토리지 풀의 속성과 이름이 요청된 모든 속성 및 풀 이름과 일치하면 Trident는 해당 스토리지 풀을 스토리지 클래스에 적합한 스토리지 풀 집합에 추가합니다. 또한 Trident는 `additionalStoragePools` 목록에 있는 모든 스토리지 풀도 해당 집합에 추가합니다. 이때 해당 스토리지 풀의 속성이 스토리지 클래스에서 요청한 속성을 전부 또는 일부 충족하지 않더라도 추가될 수 있습니다. `excludeStoragePools` 목록을 사용하여 스토리지 풀을 스토리지 클래스에서 사용하지 않도록 재정의하거나 제거할 수 있습니다. Trident는 새 백엔드를 추가할 때마다 유사한 프로세스를 수행하여 해당 백엔드의 스토리지 풀이 기존 스토리지 클래스의 스토리지 풀 요구 사항을 충족하는지 확인하고 제외로 표시된 스토리지 풀을 제거합니다.

볼륨 생성

Trident는 스토리지 클래스와 스토리지 풀 간의 연결을 사용하여 볼륨을 프로비저닝할 위치를 결정합니다. 볼륨을 생성하면 Trident는 먼저 해당 볼륨의 스토리지 클래스에 대한 스토리지 풀 집합을 가져오고, 볼륨에 프로토콜을 지정한 경우 Trident는 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 없음). Trident는 볼륨이 고르게 분산되도록 이 결과 집합의 순서를 무작위로 정렬한 다음 각 스토리지 풀에서 순차적으로 볼륨 프로비저닝을 시도합니다. 하나라도 성공하면 성공적으로 반환하고 프로세스에서 발생한 모든 실패를 로그에 기록합니다. Trident는 요청된 스토리지 클래스와 프로토콜에 사용 가능한 모든 스토리지 풀에서 프로비저닝에 실패한 경우에만 실패를 반환합니다.

볼륨 스냅샷

Trident가 드라이버에 대한 볼륨 스냅샷 생성을 처리하는 방법에 대해 자세히 알아보십시오.

볼륨 스냅샷 생성에 대해 알아보십시오

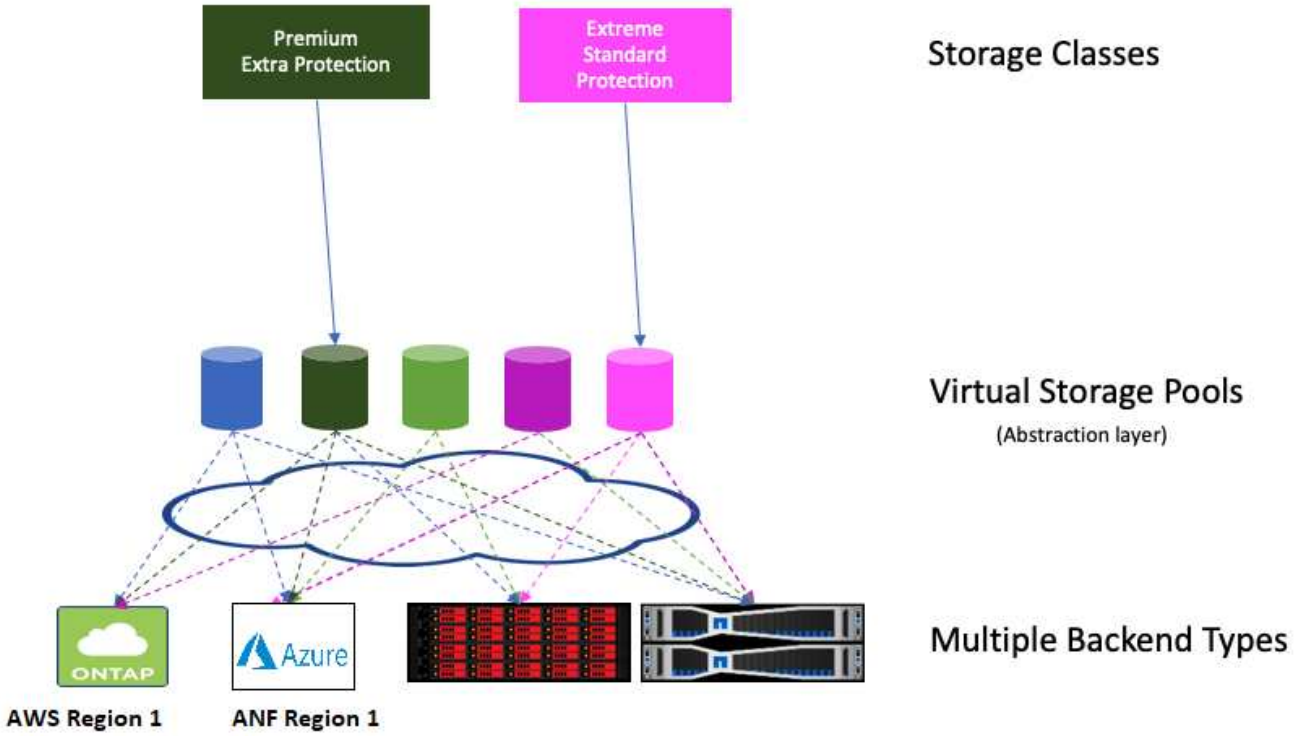
- `ontap-nas`, `ontap-san` 및 `azure-netapp-files` 드라이버의 경우 각 영구 볼륨(PV)은 FlexVol 볼륨에 매핑됩니다. 결과적으로 볼륨 스냅샷은 NetApp 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁 스냅샷 기술보다 뛰어난 안정성, 확장성, 복구 기능 및 성능을 제공합니다. 이러한 스냅샷 복사본은 생성 시간과 스토리지 공간 측면에서 매우 효율적입니다.
- `ontap-nas-flexgroup` 드라이버의 경우 각 영구 볼륨(PV)은 FlexGroup에 매핑됩니다. 결과적으로 볼륨 스냅샷은 NetApp FlexGroup 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁 스냅샷 기술보다 뛰어난 안정성, 확장성, 복구 기능 및 성능을 제공합니다. 이러한 스냅샷 복사본은 생성 시간과 스토리지 공간 측면에서 매우 효율적입니다.
- `ontap-san-economy` 드라이버의 경우, PV는 공유 FlexVol 볼륨에 생성된 LUN에 매핑됩니다. PV의 VolumeSnapshots는 연결된 LUN의 FlexClones를 수행하여 생성됩니다. ONTAP FlexClone 기술을 사용하면 가장 큰 데이터 세트의 복사본도 거의 즉시 생성할 수 있습니다. 복사본은 상위 항목과 데이터 블록을 공유하며 메타데이터에 필요한 스토리지 외에는 스토리지를 사용하지 않습니다.
- `solidfire-san` 드라이버의 경우, 각 PV는 NetApp Element 소프트웨어/NetApp HCI 클러스터에 생성된 LUN에 매핑됩니다. VolumeSnapshots은 기본 LUN의 Element 스냅샷으로 표현됩니다. 이러한 스냅샷은 특정 시점의 복사본이며 시스템 리소스와 공간을 적게 차지합니다.
- `ontap-nas` 및 `ontap-san` 드라이버를 사용할 때 ONTAP 스냅샷은 FlexVol의 시점 복사본이며 FlexVol 자체의 공간을 사용합니다. 이로 인해 스냅샷이 생성/예약됨에 따라 시간이 지남에 따라 볼륨의 쓰기 가능 공간이 줄어들 수 있습니다. 이 문제를 해결하는 간단한 방법 중 하나는 Kubernetes를 통해 크기를 조정하여 볼륨을 확장하는 것입니다. 또 다른 방법은 더 이상 필요하지 않은 스냅샷을 삭제하는 것입니다. Kubernetes를 통해 생성된 VolumeSnapshot을 삭제하면 Trident는 연결된 ONTAP 스냅샷을 삭제합니다. Kubernetes를 통해 생성되지 않은 ONTAP 스냅샷도 삭제할 수 있습니다.

Trident를 사용하면 VolumeSnapshots를 사용하여 새로운 PV를 생성할 수 있습니다. 이러한 스냅샷에서 PV를 생성하는 작업은 지원되는 ONTAP 백엔드에 대해 FlexClone 기술을 사용하여 수행됩니다. 스냅샷에서 PV를 생성할 때 백업 볼륨은 스냅샷의 상위 볼륨의 FlexClone입니다. `solidfire-san` 드라이버는 Element 소프트웨어 볼륨 클론을 사용하여 스냅샷에서 PV를 생성합니다. 여기서는 Element 스냅샷에서 클론을 생성합니다.

가상 풀은 Trident 스토리지 백엔드와 Kubernetes `StorageClasses` 사이에 추상화 계층을 제공합니다. 이를 통해 관리자는 `StorageClass` 원하는 기준을 충족하기 위해 사용할 물리적 백엔드, 백엔드 풀 또는 백엔드 유형을 지정하지 않고도 각 백엔드의 위치, 성능 및 보호와 같은 측면을 공통적이고 백엔드에 구애받지 않는 방식으로 정의할 수 있습니다.

가상 풀에 대해 알아보십시오

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



가상 풀 목록 외부에 지정된 모든 측면은 백엔드에 대해 전역적이며 모든 가상 풀에 적용되는 반면, 각 가상 풀은 하나 이상의 측면을 개별적으로 지정할 수 있습니다(백엔드 전역 측면 재정의).

- 가상 풀을 정의할 때 백엔드 정의에 있는 기존 가상 풀의 순서를 변경하려고 시도하지 마십시오.
- 기존 가상 풀의 속성을 수정하는 것은 권장하지 않습니다. 변경하려면 새 가상 풀을 정의해야 합니다.

대부분의 측면(aspect)은 백엔드별 용어로 지정됩니다. 중요한 점은 측면 값은 백엔드 드라이버 외부로 노출되지 않으며 `StorageClasses`에서 일치 검색에 사용할 수 없다는 것입니다. 대신 관리자는 각 가상 풀에 대해 하나 이상의 레이블을 정의합니다. 각 레이블은 키:값 쌍이며, 여러 백엔드에서 공통으로 사용될 수 있습니다. 측면과 마찬가지로 레이블도 풀별로 또는 백엔드 전체에 대해 지정할 수 있습니다. 미리 정의된 이름과 값을 갖는 측면과 달리, 관리자는 필요에 따라 레이블 키와 값을 자유롭게 정의할 수 있습니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

가상 풀 레이블은 다음 문자를 사용하여 정의할 수 있습니다.

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

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

운영자	예	풀의 레이블 값은 다음을 충족해야 합니다.
=	성능=프리미엄	일치
!=	성능!=extreme	일치하지 않음
in	(동쪽, 서쪽)의 위치	값 집합에 있음
notin	performance가 silver, bronze가 아님	값 집합에 포함되지 않음
<key>	보호	임의의 값으로 존재
!<key>	!보호	존재하지 않음

볼륨 액세스 그룹

Trident가 활용하는 방법에 대해 자세히 알아보세요 "볼륨 액세스 그룹".

> CHAP를 사용하는 경우 이 섹션을 무시하십시오. CHAP는 관리를 간소화하고 아래에 설명된 확장 제한을 방지하는 데 권장됩니다. 또한 CSI 모드에서 Trident를 사용하는 경우에도 이 섹션을 무시할 수 있습니다. Trident는 향상된 CSI 프로비저너로 설치될 때 CHAP를 사용합니다.

볼륨 액세스 그룹에 대해 알아보십시오

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

Trident는 구성된 액세스 그룹에 새 볼륨을 연결하지만, 액세스 그룹 자체를 생성하거나 관리하지는 않습니다. 액세스 그룹은 스토리지 백엔드를 Trident에 추가하기 전에 존재해야 하며, 해당 백엔드에서 프로비저닝된 볼륨을 마운트할 수 있는 Kubernetes 클러스터의 모든 노드에 대한 iSCSI IQN을 포함해야 합니다. 대부분의 설치 환경에서는 클러스터의 모든 워커 노드가 포함됩니다.

노드가 64개 이상인 Kubernetes 클러스터의 경우 여러 액세스 그룹을 사용해야 합니다. 각 액세스 그룹은 최대 64개의 IQN을 포함할 수 있으며 각 볼륨은 4개의 액세스 그룹에 속할 수 있습니다. 최대 4개의 액세스 그룹을 구성하면 최대 256개 노드 크기의 클러스터에 있는 모든 노드가 모든 볼륨에 액세스할 수 있습니다. 볼륨 액세스 그룹에 대한 최신 제한 사항은 "여기"을 참조하십시오.

기본 `trident` 액세스 그룹을 사용하는 구성에서 다른 액세스 그룹도 사용하는 구성으로 수정하는 경우 목록에 `trident` 액세스 그룹의 ID를 포함하십시오.

Trident 빠른 시작

Trident를 설치하고 몇 단계만 거치면 스토리지 리소스 관리를 시작할 수 있습니다. 시작하기 전에 "Trident 요구 사항"을(를) 검토하십시오.

Docker의 경우 "Docker용 Trident"을(를) 참조하십시오.

1 작업자 노드 준비

Kubernetes 클러스터의 모든 작업자 노드는 Pod용으로 프로비저닝한 볼륨을 마운트할 수 있어야 합니다.

"작업자 노드를 준비합니다"

2 **Trident** 설치

Trident는 다양한 환경과 조직에 최적화된 여러 설치 방법과 모드를 제공합니다.

"Trident 설치"

3 백엔드 생성

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. 백엔드는 Trident가 해당 스토리지 시스템과 통신하는 방법과 Trident가 해당 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

"백엔드 구성" 스토리지 시스템용

4 **Kubernetes StorageClass** 생성

Kubernetes StorageClass 객체는 Trident를 프로비저너로 지정하며, 사용자 지정 가능한 속성을 가진 볼륨을 프로비저닝하기 위한 스토리지 클래스를 생성할 수 있도록 합니다. Trident는 Trident 프로비저너를 지정하는 Kubernetes 객체에 대해 일치하는 스토리지 클래스를 생성합니다.

"스토리지 클래스를 생성합니다"

5 볼륨 프로비저닝

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

구성된 Kubernetes StorageClass를 사용하여 PersistentVolume (PV)와 PersistentVolumeClaim (PVC)를 생성하여 PV에 대한 액세스를 요청합니다. 그런 다음 PV를 파드에 마운트할 수 있습니다.

"볼륨 프로비저닝"

다음 단계

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

요구 사항

Trident를 설치하기 전에 다음 일반 시스템 요구 사항을 검토해야 합니다. 특정 백엔드에는 추가 요구 사항이 있을 수 있습니다.

Trident에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다**.**

Trident에 대한 중요 정보

- Kubernetes 1.34는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 업그레이드하십시오.
- Trident는 SAN 환경에서 다중 경로 구성 사용을 엄격하게 시행하며, multipath.conf 파일에 권장 값은 `find_multipaths: no`입니다.

  다중 경로를 사용하지 않는 구성 또는 multipath.conf 파일에서 `find_multipaths: yes` 또는 `find_multipaths: smart` 값을 사용하면 마운트가 실패합니다. Trident는 21.07 릴리스부터 `find_multipaths: no` 사용을 권장해 왔습니다.

지원되는 프런트엔드(오케스트레이터)

Trident는 다음과 같은 여러 컨테이너 엔진 및 오케스트레이터를 지원합니다.

- Anthos On-Prem(VMware) 및 bare metal용 Anthos 1.16
- Kubernetes 1.27 - 1.34
- OpenShift 4.12, 4.14 - 4.20(OpenShift 4.19에서 iSCSI 노드 준비를 사용하려는 경우 지원되는 최소 Trident 버전은 25.06.1입니다.)

> Trident는 "Red Hat Extended Update Support(EUS) 릴리스 라이프사이클"에 따라 이전 OpenShift 버전을 계속 지원합니다. 이는 업스트림에서 공식적으로 더 이상 지원되지 않는 Kubernetes 버전을 사용하는 경우에도 마찬가지입니다. 이러한 경우 Trident를 설치할 때 Kubernetes 버전에 대한 경고 메시지는 무시하셔도 됩니다.

- Rancher Kubernetes Engine 2(RKE2) v1.28.x - 1.34.x

> *Trident*는 *Rancher Kubernetes Engine 2(RKE2)* 버전 *1.27.x - 1.34.x*에서 지원되지만, *Trident*는 현재 *RKE2 v1.28.5+rke2r1*에서만 검증되었습니다.

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

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

Trident와 ONTAP는 "KubeVirt"의 스토리지 공급자로 사용할 수 있습니다.

> Trident가 설치된 Kubernetes 클러스터를 1.25에서 1.26 이상으로 업그레이드하기 전에 "Helm 설치 업그레이드"을(를) 참조하십시오.

지원되는 백엔드(스토리지)

Trident를 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 Array(ASA)
- NetApp 전체 또는 제한적 지원을 받는 ONTAP 버전을 실행하는 온프레미스 FAS, AFF 또는 ASA r2(iSCSI, NVMe/TCP 및 FC). "소프트웨어 버전 지원"을 참조하십시오.
- NetApp HCI/Element 소프트웨어 11 이상

Trident의 KubeVirt 및 OpenShift Virtualization 지원

지원되는 스토리지 드라이버:

Trident는 KubeVirt 및 OpenShift Virtualization을 위해 다음과 같은 ONTAP 드라이버를 지원합니다.

- ontap-nas
- ontap-nas-economy
- ontap-san(iSCSI, FCP, NVMe over TCP)
- ontap-san-economy(iSCSI 전용)

고려해야 할 사항:

- OpenShift Virtualization 환경에서 `fsType` 매개변수(예: `fsType: "ext4"`)를 포함하도록 스토리지 클래스를 업데이트하십시오. 필요한 경우, CDI에 Block 데이터 볼륨을 생성하도록 알리기 위해 `dataVolumeTemplates`에서 `volumeMode=Block` 매개변수를 사용하여 볼륨 모드를 명시적으로 block으로 설정하십시오.
- 블록 스토리지 드라이버의 *RWX* 액세스 모드: ontap-san(iSCSI, NVMe/TCP, FC) 및 ontap-san-economy(iSCSI) 드라이버는 "volumeMode: Block"(raw device)에서만 지원됩니다. 이러한 드라이버의 경우 볼륨이 raw device 모드로 제공되므로 `fstype` 매개변수를 사용할 수 없습니다.
- RWX 액세스 모드가 필요한 라이브 마이그레이션 워크플로의 경우 다음 조합이 지원됩니다.
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (원시 디바이스)
  - NVMe/TCP + `volumeMode=Block`(원시 디바이스)

- FC + `volumeMode=Block`(원시 디바이스)

기능 요구 사항

아래 표는 이번 Trident 릴리스에서 사용 가능한 기능과 지원하는 Kubernetes 버전을 요약한 것입니다.

기능	**Kubernetes** 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 엑스포트 정책	1.27 - 1.34	아니요
Trident Operator	1.27 - 1.34	아니요
CSI 토폴로지	1.27 - 1.34	아니요

테스트된 호스트 운영 체제

Trident는 특정 운영 체제를 공식적으로 지원하지는 않지만 다음 운영 체제에서 작동하는 것으로 알려져 있습니다.

- AMD64 및 ARM64의 OpenShift Container Platform에서 지원하는 Red Hat Enterprise Linux CoreOS(RHCOS) 버전
- AMD64 및 ARM64의 Red Hat Enterprise Linux(RHEL) 8 이상

  NVMe/TCP에는 RHEL 9 이상이 필요합니다.

- AMD64 및 ARM64의 Ubuntu 22.04 LTS 이상
- Windows Server 2022
- SUSE Linux Enterprise Server(SLES) 15 이상

기본적으로 Trident는 컨테이너에서 실행되므로 모든 Linux 워커에서 실행될 수 있습니다. 단, 사용 중인 백엔드에 따라 표준 NFS 클라이언트 또는 iSCSI 이니시에이터를 사용하여 Trident가 제공하는 볼륨을 워커에서 마운트할 수 있어야 합니다.

`tridentctl` 유틸리티는 이러한 `Linux` 배포판 모두에서 실행됩니다.

호스트 구성

Kubernetes 클러스터의 모든 워커 노드는 Pod에 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 워커 노드를 준비하려면 선택한 드라이버에 따라 NFS, iSCSI 또는 NVMe 도구를 설치해야 합니다.

"작업자 노드를 준비합니다"

스토리지 시스템 구성

Trident에서 백엔드 구성을 사용하려면 먼저 스토리지 시스템을 변경해야 할 수도 있습니다.

"백엔드 구성"

Trident 포트

Trident에는 통신을 위해 특정 포트에 대한 액세스가 필요합니다.

"Trident 포트"

컨테이너 이미지 및 해당 Kubernetes 버전

에어 갭 설치의 경우 다음 목록은 Trident를 설치하는 데 필요한 컨테이너 이미지에 대한 참조입니다. `tridentctl images` 명령을 사용하여 필요한 컨테이너 이미지 목록을 확인하십시오.

Trident 25.10에 필요한 컨테이너 이미지

Kubernetes 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:25.10.0 • docker.io/netapp/trident-autosupport:25.10 • registry.k8s.io/sig-storage/csi-provisioner:v5.3.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v1.14.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.3.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:25.10.0(선택 사항)

Trident 설치

Trident 운영자를 사용하여 설치합니다

tridentctl을 사용하여 설치

OpenShift 인증 운영자를 사용하여 설치

Trident 사용

작업자 노드를 준비합니다

Kubernetes 클러스터의 모든 워커 노드는 Pod에 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 워커 노드를 준비하려면 선택한 드라이버에 따라 NFS, iSCSI, NVMe/TCP 또는 FC 도구를 설치해야 합니다.

적절한 도구 선택

여러 드라이버를 조합해서 사용하는 경우, 해당 드라이버에 필요한 모든 도구를 설치해야 합니다. 최신 버전의 Red Hat Enterprise Linux CoreOS(RHCOS)에는 이러한 도구가 기본적으로 설치되어 있습니다.

NFS 도구

"NFS 도구를 설치합니다"다음을 사용하는 경우: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` 또는 `azure-netapp-files`.

iSCSI 도구

"iSCSI 도구를 설치합니다"다음을 사용하는 경우: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

NVMe 도구

"NVMe 툴을 설치합니다"TCP(NVMe/TCP) 프로토콜을 통한 NVMe(Nonvolatile Memory Express)에 `ontap-san`를 사용하는 경우.

> NetApp에서는 NVMe/TCP에 ONTAP 9.12 이상을 권장합니다.

FC를 통한 **SCSI** 툴

FC 및 FC-NVMe SAN 호스트 구성에 대한 자세한 내용은 "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"를 참조하십시오.

"FC 툴을 설치합니다" `ontap-san`sanType `fcp`(FC를 통한 SCSI)을 사용하는 경우.

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

SMB 도구

"SMB 볼륨 프로비저닝 준비" 다음을 사용하는 경우: `ontap-nas` SMB 볼륨을 프로비저닝합니다.

노드 서비스 검색

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

> 노드 서비스 검색은 발견된 서비스를 식별하지만 서비스가 올바르게 구성되었음을 보장하지는 않습니다. 반대로, 검색된 서비스가 없다고 해서 볼륨 마운트가 반드시 실패하는 것은 아닙니다.

이벤트 검토

Trident는 검색된 서비스를 식별하기 위해 노드에 대한 이벤트를 생성합니다. 이러한 이벤트를 검토하려면 다음을 실행합니다.

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

검색된 서비스 검토

Trident는 Trident 노드 CR에서 각 노드에 대해 활성화된 서비스를 식별합니다. 검색된 서비스를 보려면 다음을 실행합니다.

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

NFS 볼륨

운영 체제에 맞는 명령을 사용하여 NFS 툴을 설치합니다. 부팅 시 NFS 서비스가 시작되는지 확인합니다.

RHEL 8+

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

Ubuntu

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

NFS 도구를 설치한 후 워커 노드를 재부팅하여 컨테이너에 볼륨을 연결할 때 발생하는 오류를 방지하십시오.

iSCSI 볼륨

Trident는 iSCSI 세션을 자동으로 설정하고, LUN을 스캔하고, 다중 경로 디바이스를 검색하고, 포맷하고, 포드에 마운트할 수 있습니다.

iSCSI 자가 복구 기능

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

1. 원하는 iSCSI 세션 상태와 현재 iSCSI 세션 상태를 *식별*하십시오.
2. 원하는 상태와 현재 상태를 비교하여 필요한 수리를 파악합니다. Trident는 수리 우선순위와 수리를 선제적으로 진행해야 할 시점을 결정합니다.
3. 현재 iSCSI 세션 상태를 원하는 iSCSI 세션 상태로 되돌리기 위해 필요한 *복구*를 수행합니다.

자가 복구 활동에 대한 로그는 해당 Daemonset Pod의 `trident-main` 컨테이너에 있습니다. 로그를 보려면 Trident 설치 시 `debug`를 "true"로 설정해야 합니다.

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

- 네트워크 연결 문제 발생 후 발생할 수 있는 오래되었거나 불안정한 iSCSI 세션. 오래된 세션의 경우 Trident는 로그아웃하기 전에 7분 동안 기다린 후 포털과의 연결을 다시 설정합니다.

  예를 들어 스토리지 컨트롤러에서 CHAP 암호가 교체되고 네트워크 연결이 끊어지면 이전(만료된) CHAP 암호가 계속 남아 있을 수 있습니다. 자가 복구 기능은 이러한 상황을 감지하고 자동으로 세션을 재설정하여 업데이트된 CHAP 암호를 적용할 수 있습니다.

- iSCSI 세션 누락
- 누락된 LUN

Trident 업그레이드 전 고려 사항

- 노드별 igroup(23.04 이상 버전에서 도입)만 사용하는 경우 iSCSI 자체 복구 기능은 SCSI 버스의 모든 디바이스에 대해 SCSI 재검색을 시작합니다.
- 백엔드 범위 igroup(23.04부터 더 이상 사용되지 않음)만 사용 중인 경우 iSCSI 자가 복구는 SCSI 버스의 정확한 LUN ID에 대해 SCSI 재검색을 시작합니다.
- 노드별 igroup과 백엔드 범위 igroup이 혼합되어 사용되는 경우 iSCSI 자체 복구는 SCSI 버스에서 정확한 LUN ID에 대한 SCSI 재검색을 시작합니다.

iSCSI 도구를 설치합니다

운영 체제에 맞는 명령을 사용하여 iSCSI 툴을 설치합니다.

시작하기 전에

- Kubernetes 클러스터의 각 노드는 고유한 IQN을 가져야 합니다. 이는 필수 조건입니다.
- RHCOS 버전 4.5 이상 또는 기타 RHEL 호환 Linux 배포판을 `solidfire-san` 드라이버 및 Element OS 12.5 이하와 함께 사용하는 경우 `/etc/iscsi/iscsid.conf`에서 CHAP 인증 알고리즘이 MD5로 설정되어 있는지 확인하십시오. 안전한 FIPS 규격 준수 CHAP 알고리즘인 SHA1, SHA-256 및 SHA3-256은 Element 12.7에서 사용할 수 있습니다.

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

- RHEL/Red Hat Enterprise Linux CoreOS (RHCOS)를 실행하는 워커 노드에서 iSCSI PV를 사용할 때, 인라인 공간 재확보를 수행하려면 `discard` mountOption을 StorageClass에 지정하십시오. "Red Hat 문서"를 참조하십시오.
- `multipath-tools`의 최신 버전으로 업그레이드했는지 확인하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

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

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

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

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 활성화:

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

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `iscsid` 및 `multipathd`가 실행 중인지 확인하십시오.

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

6. 활성화 및 시작 `iscsi`:

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

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

```
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 확인하십시오.

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

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 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
```

> ⓘ `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `open-iscsi` 및 `multipath-tools`가 활성화되어 실행 중인지 확인하십시오.

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

> ⓘ Ubuntu 18.04의 경우, iSCSI 데몬이 시작되기 전에 `iscsiadm`를 사용하여 대상 포트를 검색해야 합니다 `open-iscsi`. 또는 `iscsi` 서비스를 수정하여 `iscsid`가 자동으로 시작되도록 할 수 있습니다.

iSCSI 자가 복구 구성 또는 비활성화

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

- **iSCSI** 자가 복구 간격: iSCSI 자가 복구가 실행되는 빈도를 결정합니다(기본값: 5분). 이 값을 작게 설정하면 더 자주 실행되도록 구성할 수 있고, 크게 설정하면 덜 자주 실행되도록 구성할 수 있습니다.

> ⓘ iSCSI 자체 복구 간격을 0으로 설정하면 iSCSI 자체 복구가 완전히 중지됩니다. iSCSI 자체 복구를 비활성화하는 것은 권장하지 않으며, iSCSI 자체 복구가 의도한 대로 작동하지 않거나 디버깅 목적으로 필요한 특정 시나리오에서만 비활성화해야 합니다.

- **iSCSI** 자가 복구 대기 시간: iSCSI 자가 복구 기능이 비정상 세션에서 로그아웃한 후 다시 로그인을 시도하기 전에 대기하는 시간을 결정합니다(기본값: 7분). 비정상으로 식별된 세션이 로그아웃된 후 다시 로그인을 시도하기 전에 더 오래 대기하도록 더 큰 값으로 구성하거나, 더 일찍 로그아웃하고 로그인하도록 더 작은 값으로 구성할 수 있습니다.

Helm

iSCSI 자체 복구 설정을 구성하거나 변경하려면 helm 설치 또는 helm 업데이트 중에 `iscsiSelfHealingInterval` 및 `iscsiSelfHealingWaitTime` 매개변수를 전달하십시오.

다음 예는 iSCSI 자체 복구 간격을 3분으로, 자체 복구 대기 시간을 6분으로 설정합니다.

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

tridentctl

iSCSI 자체 복구 설정을 구성하거나 변경하려면 tridentctl 설치 또는 업데이트 중에 `iscsi-self-healing-interval` 및 `iscsi-self-healing-wait-time` 매개변수를 전달하십시오.

다음 예는 iSCSI 자체 복구 간격을 3분으로, 자체 복구 대기 시간을 6분으로 설정합니다.

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

NVMe/TCP 볼륨

사용 중인 운영 체제에 맞는 명령을 사용하여 NVMe 툴을 설치하십시오.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 해당 커널 버전에 맞는 NVMe 패키지를 사용할 수 없는 경우 노드의 커널 버전을 NVMe 패키지가 포함된 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

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

Ubuntu

```
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 확인하십시오.

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

Trident는 `ctrl_device_tmo` 값을 수정하여 경로가 다운되더라도 NVMe가 경로를 포기하지 않도록 합니다. 이 설정을 변경하지 마십시오.

FC를 통한 SCSI 볼륨

이제 Trident와 함께 파이버 채널(FC) 프로토콜을 사용하여 ONTAP 시스템의 스토리지 리소스를 프로비저닝하고 관리할 수 있습니다.

필수 구성 요소

FC에 필요한 네트워크 및 노드 설정을 구성하십시오.

네트워크 설정

1. 대상 인터페이스의 WWPN을 가져오십시오. 자세한 내용은 "네트워크 인터페이스 show"을(를) 참조하십시오.
2. 이니시에이터(호스트)의 인터페이스에 대한 WWPN을 가져옵니다.

   해당 호스트 운영 체제 유틸리티를 참조하십시오.
3. 호스트 및 타겟의 WWPN을 사용하여 FC 스위치에서 조닝을 구성합니다.

   자세한 내용은 해당 스위치 공급업체 설명서를 참조하십시오.

   자세한 내용은 다음 ONTAP 문서를 참조하십시오.

   - "Fibre Channel 및 FCoE 조닝 개요"

- "FC 및 FC-NVMe SAN 호스트를 구성하는 방법"

FC 툴을 설치합니다

운영 체제에 대한 명령을 사용하여 FC 툴을 설치합니다.

- FC PV를 사용하는 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 워커 노드를 사용할 때, 인라인 공간 재확보를 수행하려면 `discard` StorageClass에서 mountOption을 지정하십시오. "Red Hat 문서"를 참조하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

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

2. 다중 경로 지정 활성화:

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

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipathd`이(가) 실행 중인지 확인하십시오.

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

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

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

2. 다중 경로 지정 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
```

> `/etc/multipath.conf`에 `find_multipaths no`이(가) `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipath-tools`이(가) 활성화되어 실행 중인지 확인하십시오.

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

SMB 볼륨 프로비저닝 준비

`ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다.

ONTAP 온프레미스 클러스터용 ontap-nas-economy SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 합니다. 이러한 프로토콜 중 하나라도 구성하지 않으면 SMB 볼륨 생성이 실패합니다.

`autoExportPolicy`는 SMB 볼륨에서 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음 사항을 충족해야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 비밀 키가 하나 이상 필요합니다. 비밀 키를 생성하려면 smbcreds:

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

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident가 대신 생성해 줄 수 있습니다.

   Amazon FSx for ONTAP에는 SMB 공유가 필요합니다.

   SMB 관리자 공유는 "Microsoft Management Console" 공유 폴더 스냅인을 사용하거나 ONTAP CLI를 사용하는 두 가지 방법으로 생성할 수 있습니다. ONTAP CLI를 사용하여 SMB 공유를 생성하려면 다음을 수행합니다.

   a. 필요한 경우 공유에 대한 디렉터리 경로 구조를 생성하십시오.

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

   b. 지정된 SVM과 연결된 SMB 공유를 생성합니다.

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

c. 공유가 생성되었는지 확인합니다.

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

자세한 내용은 "SMB 공유 생성"를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 "FSx for ONTAP 구성 옵션 및 예"를 참조하십시오.

매개변수	설명	예
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	`smb-share`
`nasType`	* `smb`로 설정해야 합니다.* null인 경우 기본값은 `nfs`입니다.	`smb`
`securityStyle`	새 볼륨에 대한 보안 스타일입니다. **SMB** 볼륨의 경우 **`ntfs`** 또는 **`mixed`**로 설정해야 합니다**.**	`ntfs` 또는 `mixed` SMB 볼륨의 경우
`unixPermissions`	새 볼륨의 모드입니다. **SMB** 볼륨의 경우 비워 두어야 합니다**.**	""

백엔드 구성 및 관리

백엔드 구성

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. 백엔드는 Trident가 해당 스토리지 시스템과 통신하는 방법과 Trident가 해당 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다.

Trident는 스토리지 클래스에 정의된 요구 사항과 일치하는 백엔드에서 스토리지 풀을 자동으로 제공합니다. 스토리지 시스템의 백엔드를 구성하는 방법을 알아보십시오.

- "Azure NetApp Files 백엔드 구성"
- "Google Cloud NetApp Volumes 백엔드 구성"
- "NetApp HCI 또는 SolidFire 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버로 백엔드 구성"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버로 백엔드 구성"
- "Amazon FSx for NetApp ONTAP와 함께 Trident 사용"

Azure NetApp Files

Azure NetApp Files 백엔드 구성

Azure NetApp Files를 Trident의 백엔드로 구성할 수 있습니다. Azure NetApp Files 백엔드를 사용하여 NFS 및 SMB 볼륨을 연결할 수 있습니다. Trident는 Azure Kubernetes Services(AKS) 클러스터용 관리 ID를 사용한 자격 증명 관리를 지원합니다.

Azure NetApp Files 드라이버 세부 정보

Trident는 클러스터와의 통신을 위해 다음과 같은 Azure NetApp Files 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	**volumeMode**	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 요청될 경우 Trident는 자동으로 50GiB 볼륨을 생성합니다.
- Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.

AKS용 관리 ID

Trident는 Azure Kubernetes Services 클러스터에 대한 "관리 ID"를 지원합니다. 관리형 ID에서 제공하는 간소화된 자격 증명 관리 기능을 활용하려면 다음이 필요합니다.

- AKS를 사용하여 배포된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 관리 ID
- Trident가 설치되어 있으며 `cloudProvider`를 지정하기 위한 `"Azure"`가 포함되어 있습니다.

Trident 운영자

Trident 운영자를 사용하여 Trident를 설치하려면 `tridentorchestrator_cr.yaml`을 편집하여 `cloudProvider`를 `"Azure"`로 설정하십시오. 예를 들면 다음과 같습니다.

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

Helm

다음 예시는 환경 변수 `$CP`를 사용하여 Trident sets `cloudProvider`를 Azure에 설치합니다:

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

<code>tridentctl</code>

다음 예제는 Trident를 설치하고 `cloudProvider` 플래그를 `Azure`로 설정합니다:

```
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
- `cloudProvider`을(를) 지정하고 `"Azure"` 및 `cloudIdentity`을(를) 지정하여 워크로드 ID를 지정하는 기능이 포함된 Trident가 설치되어 있습니다

Trident 운영자

Trident 운영자를 사용하여 Trident를 설치하려면 `tridentorchestrator_cr.yaml`을 편집하여 `cloudProvider`를 `"Azure"`로 설정하고 `cloudIdentity`를 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`로 설정하십시오.

예를 들면 다음과 같습니다.

```
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
```

Helm

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

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

다음 예제는 Trident를 설치하고 `cloudProvider`를 환경 변수 `$CP`를 사용하여 Azure로 설정하며 `cloudIdentity`를 환경 변수 `$CI`를 사용하여 설정합니다:

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

<code>tridentctl</code>

다음 환경 변수를 사용하여 **cloud provider** 및 **cloud identity** 플래그 값을 설정하십시오.

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

다음 예제는 Trident를 설치하고 `cloud-provider` 플래그를 `$CP`로, 그리고 `cloud-identity`를 `$CI`로 설정합니다:

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

Azure NetApp Files 백엔드 구성을 준비하세요

Azure NetApp Files 백엔드를 구성하기 전에 다음 요구 사항을 충족해야 합니다.

NFS 및 **SMB** 볼륨의 사전 요구 사항

Azure NetApp Files를 처음 사용하거나 새 위치에서 사용하는 경우 Azure NetApp Files를 설정하고 NFS 볼륨을 생성하기 위해 몇 가지 초기 구성이 필요합니다. 을 참조하십시오 "Azure: Azure NetApp Files 설정 및 NFS 볼륨 생성".

`https://azure.microsoft.com/en-us/services/netapp/["Azure NetApp Files"^]`
백엔드를 구성하고 사용하려면 다음이 필요합니다.

- `subscriptionID`, `tenantID`, `clientID`, `location` 및 `clientSecret`는 AKS 클러스터에서 관리 ID를 사용할 때 선택 사항입니다.
- `tenantID`, `clientID` 및 `clientSecret`는 AKS 클러스터에서 클라우드 ID를 사용할 때 선택 사항입니다.

- 용량 풀. "Microsoft: Azure NetApp Files용 용량 풀 생성"을 참조하십시오.
- Azure NetApp Files에 위임된 서브넷입니다. "Microsoft: Azure NetApp Files에 서브넷 위임"을 참조하십시오.
- `subscriptionID`Azure NetApp Files가 활성화된 Azure 구독에서 가져온 것입니다.
- `tenantID`, `clientID` 및 `clientSecret`는 Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory의 "앱 등록"에서 가져와야 합니다. 앱 등록은 다음 중 하나를 사용해야 합니다.
  - 소유자 또는 기여자 역할"Azure에서 미리 정의됨".
  - 구독 수준에서 "사용자 지정 Contributor 역할"(`assignableScopes`Trident에 필요한 권한으로만 제한된 다음 권한을 가진 사용자 지정 역할입니다. 사용자 지정 역할을 생성한 후 "Azure 포털을 사용하여 역할을 할당합니다".

사용자 지정 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": []
      }
    ]
  }
}
```

- `location`하나 이상의 "위임된 서브넷"를 포함하는 Azure입니다. Trident 22.01부터 `location` 매개 변수는 백엔드 구성 파일의 최상위 수준에서 필수 필드입니다. 가상 풀에 지정된 위치 값은 무시됩니다.
- `Cloud Identity`를 사용하려면, `client ID`를 "사용자 할당 관리 ID"에서 가져와서 해당 ID를 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`에 지정하십시오.

SMB 볼륨에 대한 추가 요구 사항

SMB 볼륨을 생성하려면 다음이 필요합니다.

- Active Directory가 구성되어 Azure NetApp Files에 연결되었습니다. 자세한 내용은 "Microsoft: Azure NetApp Files용 Active Directory 연결 생성 및 관리"을 참조하십시오.
- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Azure NetApp Files가 Active Directory에 인증할 수 있도록 Active Directory 자격 증명이 포함된 Trident 암호를 하나 이상 생성해야 합니다. 암호를 생성하려면 다음을 수행합니다 `smbcreds`:

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

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

Azure NetApp Files 백엔드 구성 옵션 및 예

Azure NetApp Files의 NFS 및 SMB 백엔드 구성 옵션에 대해 알아보고 구성 예제를 살펴보세요.

백엔드 **configuration** 옵션

Trident는 사용자의 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 생성합니다.

Azure NetApp Files 백엔드는 이러한 구성 옵션을 제공합니다.

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	"azure-netapp-files"
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + 임의의 문자
`subscriptionID`	Azure 구독의 구독 ID AKS 클러스터에서 관리 ID가 활성화된 경우 선택 사항입니다.	
`tenantID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 테넌트 ID는 선택 사항입니다.	
`clientID`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 클라이언트 ID는 선택 사항입니다.	
`clientSecret`	AKS 클러스터에서 관리형 ID 또는 클라우드 ID를 사용하는 경우 앱 등록의 클라이언트 암호는 선택 사항입니다.	
`serviceLevel`	`Standard`, `Premium` 또는 `Ultra` 중 하나	"" (무작위)
`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`	에 위임된 서브넷의 이름 `Microsoft.Netapp/volumes`	""

매개변수	설명	기본값
networkFeatures	볼륨에 대한 VNet 기능 집합은 Basic 또는 `Standard`일 수 있습니다. 네트워크 기능은 모든 지역에서 사용할 수 없으며 구독에서 활성화해야 할 수도 있습니다. 기능이 활성화되지 않은 상태에서 `networkFeatures`을(를) 지정하면 볼륨 프로비저닝이 실패합니다.	""
nfsMountOptions	NFS 마운트 옵션을 세밀하게 제어할 수 있습니다. SMB 볼륨에는 적용되지 않습니다. NFS 버전 4.1을 사용하여 볼륨을 마운트하려면 쉼표로 구분된 마운트 옵션 목록에 `nfsvers=4`을 포함하여 NFS v4.1을 선택합니다. 스토리지 클래스 정의에 설정된 마운트 옵션은 백엔드 구성에 설정된 마운트 옵션을 재정의합니다.	"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 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	null
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 nfs, smb 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
supportedTopologies	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하십시오.	
qosType	QoS 유형(자동 또는 수동)을 나타냅니다.	자동
maxThroughput	최대 허용 처리량을 MiB/sec 단위로 설정합니다. 수동 QoS 용량 풀에서만 지원됩니다.	4 MiB/sec

네트워크 기능에 대한 자세한 내용은 "Azure NetApp Files 볼륨에 대한 네트워크 기능 구성"을(를) 참조하십시오.

필요한 권한 및 리소스

PVC를 생성할 때 "No capacity pools found" 오류가 발생하는 경우, 앱 등록에 필요한 권한 및 리소스(서브넷, 가상 네트워크, 용량 풀)가 연결되어 있지 않을 가능성이 높습니다. 디버그 모드가 활성화된 경우 Trident는 백엔드 생성 시 검색된 Azure 리소스를 로그에 기록합니다. 적절한 역할이 사용되고 있는지 확인하십시오.

`resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` 및 `subnet`의 값은 단축 이름 또는 정규화된 이름을 사용하여 지정할 수 있습니다. 단축 이름은 동일한 이름을 가진 여러 리소스와 일치할 수 있으므로 대부분의 경우 정규화된 이름을 사용하는 것이 좋습니다.

vNet이 Azure NetApp Files(ANF) 스토리지 계정과 다른 리소스 그룹에 있는 경우 백엔드에 대한 resourceGroups 목록을 구성할 때 가상 네트워크의 리소스 그룹을 지정합니다.

`resourceGroups`, `netappAccounts` 및 `capacityPools` 값은 검색된 리소스 집합을 이 스토리지 백엔드에서 사용 가능한 리소스로 제한하는 필터이며, 어떤 조합으로든 지정할 수 있습니다. 정규화된 이름은 다음 형식을 따릅니다.

유형	형식
리소스 그룹	<resource group>
NetApp 계정	<resource group>/<netapp account>
용량 풀	<resource group>/<netapp account>/<capacity pool>
가상 네트워크	<resource group>/<virtual network>
서브넷	<resource group>/<virtual network>/<subnet>

볼륨 프로비저닝

구성 파일의 특수 섹션에서 다음 옵션을 지정하여 기본 볼륨 프로비저닝을 제어할 수 있습니다. 자세한 내용은 예시 구성을 참조하십시오.

매개변수	설명	기본값
`exportRule`	새 볼륨에 대한 내보내기 규칙입니다. `exportRule`는 CIDR 표기법으로 IPv4 주소 또는 IPv4 서브넷을 쉼표로 구분한 목록이어야 합니다. SMB 볼륨의 경우 무시됩니다.	"0.0.0.0/0"
`snapshotDir`	.snapshot 디렉터리의 표시 여부를 제어합니다	NFSv4의 경우 "true", NFSv3의 경우 "false"
`size`	새 볼륨의 기본 크기	"100G"
`unixPermissions`	새 볼륨의 unix 권한(8진수 4자리). SMB 볼륨의 경우 무시됩니다.	"" (미리보기 기능이며, 구독 시 화이트리스트 등록이 필요합니다.)

예시 구성

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 절대 최소 백엔드 구성입니다. 이 구성을 사용하면 Trident는 구성된 위치에서 Azure NetApp Files에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀과 서브넷 중 하나에 새 볼륨을 무작위로 배치합니다. `nasType`이(가) 생략되었으므로 `nfs` 기본값이 적용되어 백엔드에서 NFS 볼륨을 프로비저닝합니다.

이 구성은 Azure NetApp Files를 처음 시작하고 기능을 테스트할 때 이상적이지만, 실제로는 프로비저닝하는 볼륨에 대한 추가 범위 지정이 필요할 것입니다.

```
---
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
```

AKS용 관리 ID

이 백엔드 구성에서는 관리 ID를 사용할 때 선택 사항인 `subscriptionID`, `tenantID`, `clientID` 및 `clientSecret`가 생략되었습니다.

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

AKS용 클라우드 ID

이 백엔드 구성은 클라우드 ID를 사용할 때 선택 사항인 `tenantID`, `clientID` 및 `clientSecret`를 생략합니다.

```
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 `eastus` 위치에 있는 `Ultra` 용량 풀에 볼륨을 배치합니다. Trident는 해당 위치에서 Azure NetApp Files에 위임된 모든 서브넷을 자동으로 검색하고 그중 하나에 새 볼륨을 임의로 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 예

이 백엔드 구성은 수동 QoS 용량 풀을 사용하여 Azure의 `eastus` 위치에 볼륨을 배치합니다.

```
---
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 축소하고 일부 볼륨 프로비저닝 기본값을 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: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/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"
```

가상 풀 구성

이 백엔드 구성은 단일 파일에 여러 스토리지 풀을 정의합니다. 이는 서로 다른 서비스 수준을 지원하는 여러 용량 풀이 있고 이를 나타내는 스토리지 클래스를 Kubernetes에 생성하려는 경우에 유용합니다. 가상 풀 레이블은 `performance`을 기반으로 풀을 구분하는 데 사용되었습니다.

```
---
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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

지원되는 토폴로지 구성

Trident는 리전 및 가용 영역을 기반으로 워크로드용 볼륨 프로비저닝을 지원합니다. 이 백엔드 구성의 `supportedTopologies` 블록은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 리전 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 리전 및 영역 값과 일치해야 합니다. 이러한 리전 및 영역은 스토리지 클래스에 제공될 수 있는 허용 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우 Trident는 언급된 리전 및 영역에 볼륨을 생성합니다. 자세한 내용은 "CSI 토폴로지 사용"를 참조하십시오.

```
---
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 스토리지 풀을 참조합니다.

`parameter.selector` 필드를 사용한 예시 정의

`parameter.selector`을 사용하면 각 `StorageClass`에 대해 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 측면을 갖게 됩니다.

```
---
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 정의 예

```
`nasType`, `node-stage-secret-name` 및 `node-stage-secret-namespace`를 사용하면 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 서로 다른 시크릿 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}
```

볼륨마다 다른 시크릿 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 필터입니다.

백엔드 생성

백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 식별하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes 백엔드 구성

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

Google Cloud NetApp Volumes 드라이버 세부 정보

Trident는 `google-cloud-netapp-volumes` 드라이버를 제공하여 클러스터와 통신합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`google-cloud-netapp-volumes`	NFS SMB	파일 시스템	RWO, ROX, RWX, RWOP	`nfs`, `smb`

GKE용 클라우드 ID

Cloud ID를 사용하면 Kubernetes Pod가 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 워크로드 ID가 구성되었고 노드 풀에 GKE MetaData Server가 구성되었습니다.
- Google Cloud NetApp Volumes 관리자(roles/netapp.admin) 역할 또는 사용자 지정 역할이 있는 GCP 서비스 계정.

- Trident 설치에는 cloudProvider를 지정하여 "GCP"를 지정하고 cloudIdentity를 지정하여 새 GCP 서비스 계정을 지정하는 것이 포함됩니다. 아래에 예시가 나와 있습니다.

Trident 운영자

Trident 운영자를 사용하여 Trident를 설치하려면 `tridentorchestrator_cr.yaml`을 편집하여 `cloudProvider`를 `"GCP"`로 설정하고 `cloudIdentity`를 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`로 설정하십시오.

예를 들면 다음과 같습니다.

```
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'
```

Helm

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

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

다음 예시는 Trident를 설치하고 환경 변수 `$CP`를 사용하여 `cloudProvider`를 GCP로 설정하며, 환경 변수 `$ANNOTATION`를 사용하여 `cloudIdentity`를 설정합니다:

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

<code>tridentctl</code>

다음 환경 변수를 사용하여 **cloud provider** 및 **cloud identity** 플래그 값을 설정하십시오.

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

다음 예제는 Trident를 설치하고 `cloud-provider` 플래그를 `$CP`로, 그리고 `cloud-identity`를 `$ANNOTATION`로 설정합니다:

```
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 사항을 충족해야 합니다.

NFS 볼륨의 사전 요구 사항

Google Cloud NetApp Volumes를 처음 사용하거나 새 위치에서 사용하는 경우 Google Cloud NetApp Volumes를 설정하고 NFS 볼륨을 생성하기 위해 몇 가지 초기 구성이 필요합니다. "시작하기 전에"을 참조하십시오.

Google Cloud NetApp Volumes 백엔드를 구성하기 전에 다음 사항을 확인하십시오.

- Google Cloud NetApp Volumes 서비스로 구성된 Google Cloud 계정. "Google Cloud NetApp Volumes"을 참조하십시오.
- Google Cloud 계정의 프로젝트 번호입니다. "프로젝트 식별"을(를) 참조하십시오.
- NetApp Volumes Admin (`roles/netapp.admin` 역할을 가진 Google Cloud 서비스 계정 "Identity and Access Management 역할 및 권한"을 참조하십시오.
- GCNV 계정용 API 키 파일입니다. 다음을 참조하십시오. "서비스 계정 키 생성"
- 스토리지 풀. "스토리지 풀 개요"을(를) 참조하십시오.

Google Cloud NetApp Volumes에 대한 액세스 설정 방법에 대한 자세한 내용은 "Google Cloud NetApp Volumes에 대한 액세스 설정"을 참조하십시오.

Google Cloud NetApp Volumes 백엔드 구성 옵션 및 예시

Google Cloud NetApp Volumes의 백엔드 구성 옵션에 대해 알아보고 구성 예제를 검토하십시오.

백엔드 configuration 옵션

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

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	`storageDriverName`의 값은 "google-cloud-netapp-volumes"로 지정해야 합니다.
`backendName`	(선택 사항) 스토리지 백엔드의 사용자 지정 이름	드라이버 이름 + "_" + API key의 일부
`storagePools`	볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개 변수입니다.	
`projectNumber`	Google Cloud 계정 프로젝트 번호입니다. 이 값은 Google Cloud 포털 홈페이지에서 확인할 수 있습니다.	

매개변수	설명	기본값
location	Trident가 GCNV 볼륨을 생성하는 Google Cloud 위치입니다. 교차 리전 Kubernetes 클러스터를 생성할 때 `location`에서 생성된 볼륨은 여러 Google Cloud 리전의 노드에서 예약된 워크로드에 사용할 수 있습니다. 교차 리전 트래픽에는 추가 비용이 발생합니다.	
apiKey	netapp.admin 역할이 있는 Google Cloud 서비스 계정의 API 키입니다. 여기에는 Google Cloud 서비스 계정의 개인 키 파일의 JSON 형식 내용이 포함됩니다(백엔드 구성 파일에 그대로 복사됨). apiKey`에는 다음 키에 대한 키-값 쌍이 포함되어야 합니다: `type, project_id, client_email, client_id, auth_uri, token_uri, auth_provider_x509_cert_url 및 client_x509_cert_url.	
nfsMountOptions	NFS 마운트 옵션에 대한 세밀한 제어	"nfsvers=3"
limitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝이 실패합니다.	"" (기본적으로 적용되지 않음)
serviceLevel	스토리지 풀 및 해당 볼륨의 서비스 수준입니다. 값은 flex, standard, premium 또는 `extreme`입니다.	
labels	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
network	GCNV 볼륨에 사용되는 Google Cloud 네트워크입니다.	
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예 {"api":false, "method":true}. 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	null
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 nfs, smb 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
supportedTopologies	이 백엔드에서 지원하는 지역 및 영역 목록을 나타냅니다. 자세한 내용은 "CSI 토폴로지 사용"을 참조하세요. 예를 들면 다음과 같습니다. supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

볼륨 프로비저닝 옵션

구성 파일의 defaults 섹션에서 기본 볼륨 프로비저닝을 제어할 수 있습니다.

매개변수	설명	기본값
exportRule	새 볼륨에 대한 내보내기 규칙입니다. IPv4 주소의 모든 조합을 쉼표로 구분한 목록이어야 합니다.	"0.0.0.0/0"

매개변수	설명	기본값
`snapshotDir`	`.snapshot` 디렉토리에 대한 액세스	NFSv4의 경우 "true", NFSv3의 경우 "false"
`snapshotReserve`	스냅샷용으로 예약된 볼륨의 비율	""(기본값 0 적용)
`unixPermissions`	새 볼륨의 Unix 권한(8진수 4자리).	""

예시 구성

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

최소 구성

이는 최소한의 백엔드 구성입니다. 이 구성을 사용하면 Trident는 구성된 위치에서 Google Cloud NetApp Volumes에 위임된 모든 스토리지 풀을 검색하고 새 볼륨을 해당 풀 중 하나에 무작위로 배치합니다. `nasType`이 생략되었으므로 `nfs` 기본값이 적용되어 백엔드에서 NFS 볼륨을 프로비저닝합니다.

이 구성은 Google Cloud NetApp Volumes를 처음 사용하고 테스트할 때 이상적이지만, 실제로는 프로비저닝하는 볼륨에 대해 추가적인 범위 지정이 필요할 가능성이 높습니다.

```
---
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
  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 `storage` 섹션에 정의됩니다. 여러 개의 스토리지 풀이 서로 다른 서비스 수준을 지원하고, 이를 나타내는 스토리지 클래스를 Kubernetes에 생성하려는 경우에 유용합니다. 가상 풀 레이블은 풀을 구분하는 데 사용됩니다. 예를 들어, 아래 예시에서는 `performance` 레이블과 `serviceLevel` 유형을 사용하여 가상 풀을 구분합니다.

모든 가상 풀에 적용할 기본값을 설정하고 개별 가상 풀의 기본값을 재정의할 수도 있습니다. 다음 예에서는 `snapshotReserve`와 `exportRule`가 모든 가상 풀의 기본값으로 사용됩니다.

자세한 내용은 "가상 풀"을(를) 참조하십시오.

```
---
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

Trident는 리전 및 가용 영역을 기반으로 워크로드용 볼륨 프로비저닝을 지원합니다. 이 백엔드 구성의 `supportedTopologies` 블록은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정된 리전 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 리전 및 영역 값과 일치해야 합니다. 이러한 리전 및 영역은 스토리지 클래스에 제공될 수 있는 허용 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 스토리지 클래스의 경우 Trident는 언급된 리전 및 영역에 볼륨을 생성합니다. 자세한 내용은 "CSI 토폴로지 사용"를 참조하십시오.

```
---
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
```

다음 단계

백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

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

백엔드가 성공적으로 생성되었는지 확인하려면 다음 명령을 실행하십시오.

```
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. `kubectl get tridentbackendconfig <backend-name>` 명령을 사용하여 백엔드를 설명하거나 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 식별하고 수정한 후 백엔드를 삭제하고 생성 명령을 다시 실행할 수 있습니다.

스토리지 클래스 정의

다음은 위의 백엔드를 참조하는 기본 `StorageClass` 정의입니다.

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

- `parameter.selector` 필드를 사용한 예시 정의:*

  `parameter.selector`을 사용하면 각 `StorageClass`에 대해 볼륨을 호스팅하는 데 사용되는 xref:{relative_path}../trident-concepts/virtual-storage-pool.html["가상 풀"]을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 속성을 갖게 됩니다.

```
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
```

스토리지 클래스에 대한 자세한 내용은 "스토리지 클래스를 생성합니다"을(를) 참조하십시오.

SMB 볼륨에 대한 정의 예

`nasType`, `node-stage-secret-name` 및 `node-stage-secret-namespace`를 사용하여 SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다. 노드 단계 비밀에는 권한이 있거나 없는 모든 Active Directory 사용자/암호를 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 서로 다른 시크릿 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}
```

볼륨마다 다른 시크릿 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 필터입니다.

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 실행합니다.

```
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
```

NetApp HCI 또는 SolidFire 백엔드 구성

Trident 설치 환경에서 Element 백엔드를 생성하고 사용하는 방법을 알아보십시오.

Element 드라이버 세부 정보

Trident는 `solidfire-san` 스토리지 드라이버를 제공하여 클러스터와 통신합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

```
`solidfire-san` 스토리지 드라이버는 _file_ 및 _block_ 볼륨 모드를 지원합니다.
`Filesystem` volumeMode의 경우 Trident는 볼륨을 생성하고 파일 시스템을 생성합니다.
파일 시스템 유형은 StorageClass에 의해 지정됩니다.
```

드라이버	프로토콜	**VolumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`solidfire-san`	iSCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템이 없습니다. 원시 블록 장치입니다.
`solidfire-san`	iSCSI	파일 시스템	RWO, RWOP	`xfs`, `ext3`, `ext4`

시작하기 전에

Element 백엔드를 생성하기 전에 다음 사항이 필요합니다.

- Element 소프트웨어를 실행하는 지원되는 스토리지 시스템.
- 볼륨을 관리할 수 있는 NetApp HCI/SolidFire 클러스터 관리자 또는 테넌트 사용자의 자격 증명입니다.
- 모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. "작업자 노드 준비 정보"을(를) 참조하십시오.

백엔드 **configuration** 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	항상 "solidfire-san"
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	"solidfire_" + 스토리지(iSCSI) IP 주소
`Endpoint`	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP	
`SVIP`	스토리지(iSCSI) IP 주소 및 포트	
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다.	""
`TenantName`	사용할 테넌트 이름(찾을 수 없는 경우 생성됨)	
`InitiatorIFace`	iSCSI 트래픽을 특정 호스트 인터페이스로 제한합니다	"default"
`UseCHAP`	iSCSI 인증에는 CHAP를 사용합니다. Trident는 CHAP를 사용합니다.	true
`AccessGroups`	사용할 액세스 그룹 ID 목록	"Trident"라는 이름의 액세스 그룹 ID를 찾습니다.
`Types`	QoS 사양	
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다	"" (기본적으로 적용되지 않음)

매개변수	설명	기본값
debugTraceFlags	문제 해결 시 사용할 디버그 플래그입니다. 예를 들어, {"api":false, "method":true}	null

`debugTraceFlags`문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.

예시 **1:** 세 가지 볼륨 유형을 사용하는 `solidfire-san` 드라이버의 백엔드 구성

이 예제는 CHAP 인증을 사용하는 백엔드 파일을 보여주며, 특정 QoS 보장을 적용한 세 가지 볼륨 유형을 모델링합니다. 일반적으로는 `IOPS` 스토리지 클래스 매개변수를 사용하여 이러한 각 볼륨 유형을 사용할 스토리지 클래스를 정의하게 됩니다.

```
---
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
```

예제 **2:** 가상 풀을 사용하는 `solidfire-san` 드라이버의 백엔드 및 스토리지 클래스 구성

이 예시는 가상 풀과 해당 가상 풀을 참조하는 StorageClasses로 구성된 백엔드 정의 파일을 보여줍니다.

Trident는 프로비저닝 시 스토리지 풀에 있는 레이블을 백엔드 스토리지 LUN으로 복사합니다. 편의를 위해 스토리지

관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

아래 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값이 설정되어 있으며, 이는 `type`을 `Silver`로 설정합니다. 가상 풀은 `storage` 섹션에서 정의됩니다. 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. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

첫 번째 StorageClass(`solidfire-gold-four`는 첫 번째 가상 풀에 매핑됩니다. 이 풀은 Gold `Volume Type QoS`의 Gold 성능을 제공하는 유일한 풀입니다. 마지막 StorageClass(`solidfire-silver`는 Silver 성능을 제공하는 모든 스토리지 풀을 지정합니다. Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 합니다.

```
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
```

자세한 정보 찾기

- "볼륨 액세스 그룹"

ONTAP SAN 드라이버

ONTAP SAN 드라이버 개요

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

ONTAP SAN 드라이버 세부 정보

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

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	FC를 통한 iSCSI SCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스

드라이버	프로토콜	**volumeMode**	지원되는 액세스 모드	지원되는 파일 시스템
`ontap-san`	FC를 통한 iSCSI SCSI	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP NVMe/TCP에 대한 추가 고려 사항을 참조하십시오.	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스
`ontap-san`	NVMe/TCP NVMe/TCP에 대한 추가 고려 사항을 참조하십시오.	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	블록	RWO, ROX, RWX, RWOP	파일 시스템 없음, 원시 블록 디바이스
`ontap-san-economy`	iSCSI	파일 시스템	RWO, RWOP ROX 및 RWX는 Filesystem 볼륨 모드에서 사용할 수 없습니다.	`xfs`, `ext3`, `ext4`

- `ontap-san-economy`지속적 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되는 경우에만 사용하십시오.
- `ontap-nas-economy`영구 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되고 `ontap-san-economy` 드라이버를 사용할 수 없는 경우에만 사용하십시오.
- 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우에는 `ontap-nas-economy`사용하지 마십시오.
- NetApp은 ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 확장을 사용하지 않는 것을 권장합니다. 해결 방법으로 Trident는 스냅샷 예약 기능을 지원하며, 이에 따라 Flexvol 볼륨을 확장합니다.

사용자 권한

Trident는 일반적으로 `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다. Amazon FSx for NetApp ONTAP 배포의 경우 Trident는 클러스터 `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자, 또는 동일한 역할을 가진 다른 이름의 사용자를

사용하여 ONTAP 또는 SVM 관리자로 실행되어야 합니다. `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

> `limitAggregateUsage` 매개변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Trident와 함께 Amazon FSx for NetApp ONTAP를 사용하는 경우 `limitAggregateUsage` 매개변수는 `vsadmin` 및 `fsxadmin` 사용자 계정에서 작동하지 않습니다. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 더욱 제한적인 역할을 생성하는 것도 가능하지만 권장하지 않습니다. 대부분의 새로운 Trident 릴리스에서는 추가 API를 호출하므로 이를 고려해야 하기 때문에 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

NVMe/TCP에 대한 추가 고려 사항

Trident는 `ontap-san` 드라이버를 사용하여 다음을 포함한 NVMe(Non-Volatile Memory Express) 프로토콜을 지원합니다.

- IPv6
- NVMe 볼륨의 스냅샷 및 클론
- NVMe 볼륨 크기 조정
- Trident 외부에서 생성된 NVMe 볼륨을 가져와서 Trident에서 수명 주기를 관리할 수 있도록 합니다
- NVMe 네이티브 다중 경로
- K8s 노드의 정상 종료 또는 비정상 종료(24.06)

Trident는 다음을 지원하지 않습니다.

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

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

ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오

ONTAP SAN 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항 및 인증 옵션을 이해하십시오.

요구 사항

모든 ONTAP 백엔드의 경우 Trident를 사용하려면 SVM에 하나 이상의 애그리게이트를 할당해야 합니다.

> "ASA r2 시스템"는 스토리지 계층 구현 방식에서 다른 ONTAP 시스템(ASA, AFF, FAS)과 차이가 있습니다. ASA r2 시스템에서는 애그리게이트 대신 스토리지 가용성 영역을 사용합니다. ASA r2 시스템에서 SVM에 애그리게이트를 할당하는 방법에 대한 "이것" 기술 자료 문서를 참조하십시오.

여러 드라이버를 실행할 수도 있고, 특정 드라이버를 가리키는 스토리지 클래스를 생성할 수도 있다는 점을 기억하세요. 예를 들어, `san-dev` 드라이버를 사용하는 `ontap-san` 클래스와 `san-default` 드라이버를 사용하는 `ontap-`

`san-economy` 클래스를 구성할 수 있습니다.

모든 Kubernetes 워커 노드에는 적절한 iSCSI 도구가 설치되어 있어야 합니다. 자세한 내용은 "작업자 노드를 준비합니다"을 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한을 가진 ONTAP 사용자의 사용자 이름과 암호입니다. ONTAP 버전과의 최대 호환성을 보장하기 위해 `admin` 또는 `vsadmin`와 같이 미리 정의된 보안 로그인 역할을 사용하는 것이 좋습니다.
- 인증서 기반: Trident는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키, 그리고 사용하는 경우(권장) 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값이 포함되어야 합니다.

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

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

자격 증명 기반 인증 활성화

Trident는 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. `admin` 또는 `vsadmin`와 같은 표준 사전 정의된 역할을 사용하는 것이 좋습니다. 이렇게 하면 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와의 상위 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 생성하여 Trident와 함께 사용할 수 있지만 권장하지 않습니다.

백엔드 정의 샘플은 다음과 같습니다.

YAML

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

JSON

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

백엔드 정의는 자격 증명이 평문으로 저장되는 유일한 위치라는 점을 명심하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 시크릿으로 저장됩니다. 백엔드 생성 또는 업데이트는 자격 증명을 알아야 하는 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행하는 관리자 전용 작업입니다.

인증서 기반 인증을 활성화합니다

신규 및 기존 백엔드는 인증서를 사용하여 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 가지 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64 인코딩 값입니다.
- clientPrivateKey: 연결된 개인 키의 Base64 인코딩 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 반드시 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않는 경우에는 이 매개변수를 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성 시 CN(일반 이름)을 인증할 ONTAP 사용자로 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. 스토리지 관리자가 이미 처리했을 수 있습니다. 신뢰할 수 있는 CA를 사용하지 않는 경우 이 단계를 건너뛰십시오.

```
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 인증서 및 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
```

ⓘ 이 명령을 실행하면 ONTAP에서 인증서 입력을 요청합니다. 1단계에서 생성된 `k8senv.pem` 파일의 내용을 붙여넣은 다음 `END`를 입력하여 설치를 완료하십시오.

4. ONTAP 보안 로그인 역할이 `cert` 인증 방법을 지원하는지 확인합니다.

```
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 테스트하십시오. <ONTAP Management LIF> 및 <vserver name>를 Management LIF IP 및 SVM 이름으로 바꾸십시오.

```
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. 인증서, 키 및 신뢰할 수 있는 CA 인증서를 Base64로 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 방법을 업데이트하거나 자격 증명을 변경하세요

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 교체할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하는 방식으로 업데이트할 수 있고, 인증서를 사용하는 백엔드를 사용자 이름/암호 기반으로 업데이트할 수도 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수가 포함된 업데이트된 backend.json 파일을 사용하여 `tridentctl backend update`을(를) 실행합니다.

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

암호를 교체할 때는 스토리지 관리자가 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 후 백엔드 업데이트를 진행합니다. 인증서를 교체할 때는 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 백엔드를 업데이트하여 새 인증서를 사용하도록 설정한 후, ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 생성된 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공적으로 완료되면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 의미합니다.

Trident용 사용자 지정 ONTAP 역할 생성

Trident에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 권한으로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident는 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 "Trident 사용자 지정 역할 생성기"을 참조하십시오.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 생성합니다.

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

2. Trident 사용자의 사용자 이름을 생성합니다.

```
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 역할 매핑:

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

System Manager 사용

ONTAP System Manager에서 다음 단계를 수행하십시오.

1. 사용자 지정 역할 생성:
   a. 클러스터 수준에서 사용자 지정 역할을 생성하려면 * Cluster > Settings * 를 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 생성하려면 *스토리지 > 스토리지 VM > `required SVM`> 설정 > 사용자 및 역할*을 선택하십시오.

   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 아래에서 추가 아이콘 *+*를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

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

양방향 **CHAP**를 사용하여 연결을 인증합니다

Trident는 `ontap-san` 및 `ontap-san-economy` 드라이버에 대해 양방향 CHAP를 사용하여 iSCSI 세션을 인증할 수 있습니다. 이를 위해서는 백엔드 정의에서 `useCHAP` 옵션을 활성화해야 합니다. `true`로 설정하면 Trident는 SVM의 기본 이니시에이터 보안을 양방향 CHAP로 구성하고 백엔드 파일에서 사용자 이름과 암호를 설정합니다.

NetApp은 연결 인증에 양방향 CHAP를 사용하는 것을 권장합니다. 다음 샘플 구성을 참조하십시오.

```
---
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
```

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

`useCHAP=true` 외에도 `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` 및 `chapUsername` 필드는 백엔드 정의에 반드시 포함되어야 합니다. 백엔드 생성 후에는 `tridentctl update`를 실행하여 비밀 키를 변경할 수 있습니다.

작동 방식

`useCHAP`을 `true`로 설정하면 스토리지 관리자가 `Trident`에 스토리지 백엔드에서 `CHAP`를 구성하도록 지시합니다. 여기에는 다음 사항이 포함됩니다.

- SVM에서 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 없음(기본값으로 설정)이고 볼륨에 기존 LUN이 없는 경우, Trident는 기본 보안 유형을 `CHAP`로 설정하고 CHAP 이니시에이터 및 대상 사용자 이름과 암호 구성을 진행합니다.
  - SVM에 LUN이 포함되어 있는 경우 Trident는 SVM에서 CHAP를 활성화하지 않습니다. 이렇게 하면 SVM에 이미 있는 LUN에 대한 액세스가 제한되지 않습니다.
- CHAP 이니시에이터 및 타겟 사용자 이름과 암호를 구성합니다. 이러한 옵션은 백엔드 구성에 지정해야 합니다(위 참조).

백엔드가 생성되면 Trident는 해당 `tridentbackend` CRD를 생성하고 CHAP 시크릿과 사용자 이름을 Kubernetes 시크릿으로 저장합니다. 이 백엔드에서 Trident가 생성하는 모든 PV는 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 교체하고 백엔드를 업데이트합니다

`backend.json` 파일의 CHAP 매개변수를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있습니다. 이를 위해서는 CHAP 암호를 업데이트하고 `tridentctl update` 명령을 사용하여 이러한 변경 사항을 반영해야 합니다.

백엔드의 CHAP 암호를 업데이트할 때는 `tridentctl`을 사용하여 백엔드를 업데이트해야 합니다. Trident가 이러한 변경 사항을 인식할 수 없으므로 ONTAP CLI 또는 ONTAP System Manager를 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 Trident가 자격 증명을 업데이트하면 계속 활성 상태로 유지됩니다. 새 연결은 업데이트된 자격 증명을 사용하고 기존 연결은 계속 활성 상태로 유지됩니다. 기존 PV를 연결 해제했다가 다시 연결하면 업데이트된 자격 증명이 사용됩니다.

ONTAP SAN 구성 옵션 및 예

Trident 설치 환경에서 ONTAP SAN 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를

제공합니다.

"ASA r2 시스템" 는 스토리지 계층 구현에서 다른 ONTAP 시스템(ASA, AFF 및 FAS)과 다릅니다. 이러한 차이점은 표기된 특정 매개변수의 사용에 영향을 미칩니다. "ASA r2 시스템과 다른 ONTAP 시스템 간의 차이점에 대해 자세히 알아보십시오".

ontap-san 드라이버(iSCSI, NVMe/TCP 및 FC 프로토콜 포함)만 ASA r2 시스템에서 지원됩니다.

Trident 백엔드 구성에서 시스템이 ASA r2임을 지정할 필요가 없습니다. `ontap-san`을 `storageDriverName`로 선택하면 Trident가 ASA r2 또는 기타 ONTAP 시스템을 자동으로 감지합니다. 아래 표에 나와 있는 것처럼 일부 백엔드 구성 매개 변수는 ASA r2 시스템에 적용되지 않습니다.

백엔드 **configuration** 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값
version		항상 1
storageDriverName	스토리지 드라이버의 이름	ontap-san 또는 ontap-san-economy
backendName	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
managementLIF	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. 원활한 MetroCluster 전환을 위해서는 MetroCluster 예시을 참조하십시오. "vsadmin" 자격 증명을 사용하는 경우 `managementLIF`은 SVM의 자격 증명이어야 하며, "admin" 자격 증명을 사용하는 경우 `managementLIF`는 클러스터의 자격 증명이어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"
dataLIF	프로토콜 LIF의 IP 주소입니다. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. **iSCSI**의 경우 지정하지 마십시오. Trident는 "ONTAP 선택적 LUN 매핑"를 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. `dataLIF`가 명시적으로 정의된 경우 경고가 생성됩니다. **MetroCluster**의 경우 생략합니다. MetroCluster 예시를 참조하십시오.	SVM에 의해 도출됨

매개변수	설명	기본값
`svm`	사용할 스토리지 가상 머신 **MetroCluster**의 경우 생략 MetroCluster 예시을 참조하십시오.	SVM `managementLIF`이 지정된 경우 파생됩니다
`useCHAP`	ONTAP SAN 드라이버용 iSCSI 인증에 CHAP를 사용합니다[부울 매개 변수]. 백엔드에 제공된 SVM에 대해 양방향 CHAP를 기본 인증으로 구성하고 사용하려면 Trident에 대해 `true`로 설정하십시오. 자세한 내용은 "ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오"를 참조하십시오. **FCP** 또는 **NVMe/TCP**에서는 지원되지 않습니다.	`false`
`chapInitiatorSecret`	CHAP 이니시에이터 암호입니다. 다음과 같은 경우 필수 항목입니다 `useCHAP=true`	""
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`chapTargetInitiatorSecret`	CHAP 대상 개시자 비밀 키. 다음 경우 필수 항목입니다 `useCHAP=true`	""
`chapUsername`	인바운드 사용자 이름. 다음 조건에서 필수 항목입니다 `useCHAP=true`	""
`chapTargetUsername`	대상 사용자 이름. 다음의 경우 필수 사항입니다 `useCHAP=true`	""
`clientCertificate`	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
`clientPrivateKey`	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다.	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다.	""
`username`	ONTAP 클러스터와 통신하는 데 필요한 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대한 자세한 내용은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을 참조하십시오.	""
`password`	ONTAP 클러스터와 통신하는 데 필요한 암호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	""
`svm`	사용할 스토리지 가상 머신	SVM `managementLIF`이 지정된 경우 파생됩니다
`storagePrefix`	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 나중에 수정할 수 없습니다. 이 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`trident`

매개변수	설명	기본값
`aggregate`	프로비저닝용 애그리게이트(선택 사항, 설정된 경우 SVM에 할당해야 함). `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당하지 않으면 사용 가능한 애그리게이트 중 하나를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident에서 백엔드가 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **ASA r2** 시스템에는 지정하지 마십시오.	""
`limitAggregateUsage`	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. Amazon FSx for NetApp ONTAP 백엔드를 사용하는 경우 `limitAggregateUsage`을(를) 지정하지 마십시오. 제공된 `fsxadmin` 및 `vsadmin`에는 애그리게이트 사용량을 검색하고 Trident를 사용하여 제한하는 데 필요한 권한이 포함되어 있지 않습니다. **ASA r2** 시스템에는 지정하지 마십시오.	"" (기본적으로 적용되지 않음)
`limitVolumeSize`	요청된 볼륨 크기가 이 값을 초과하면 프로비저닝에 실패합니다. 또한 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	FlexVol당 최대 LUN 수는 [50, 200] 범위 내에 있어야 합니다.	`100`
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	`null`

매개변수	설명	기본값
useREST	ONTAP REST API를 사용하는 부울 매개 변수입니다. `useREST`로 설정하면 `true` Trident는 ONTAP REST API를 사용하여 백엔드와 통신하고, `false`로 설정하면 Trident는 ONTAPI(ZAPI) 호출을 사용하여 백엔드와 통신합니다. 이 기능은 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontapi` 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 `vsadmin` 및 `cluster-admin` 역할로 충족됩니다. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST`는 기본적으로 `true`로 설정되며, ONTAPI(ZAPI) 호출을 사용하려면 `useREST`를 `false`로 변경하십시오. `useREST`는 NVMe/TCP에 대한 모든 자격을 갖추고 있습니다. NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다. 지정된 경우 **ASA r2** 시스템의 경우 항상 **`true`**로 설정하십시오.	ONTAP 9.15.1 이상의 경우 true, 그렇지 않은 경우 false.
sanType	iSCSI의 경우 iscsi, NVMe/TCP의 경우 nvme 또는 Fibre Channel(FC)을 통한 SCSI의 경우 `fcp`를 선택하는 데 사용합니다.	iscsi 공백인 경우
formatOptions	`formatOptions`를 사용하여 `mkfs` 명령에 대한 명령줄 인수를 지정할 수 있으며, 이는 볼륨을 포맷할 때마다 적용됩니다. 이를 통해 원하는 대로 볼륨을 포맷할 수 있습니다. 장치 경로를 제외하고 mkfs 명령 옵션과 유사하게 formatOptions를 지정해야 합니다. 예: "-E nodiscard" • ontap-san 및 ontap-san-economy 드라이버에서 iSCSI 프로토콜과 함께 지원됩니다.* 또한 **iSCSI** 및 **NVMe/TCP** 프로토콜을 사용하는 **ASA r2** 시스템에서도 지원됩니다.	

매개변수	설명	기본값
`limitVolumePoolSize`	ontap-san-economy 백엔드에서 LUN을 사용할 때 요청할 수 있는 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)
`denyNewVolumePools`	`ontap-san-economy` 백엔드에서 LUN을 포함할 새 FlexVol 볼륨 생성을 제한합니다. 기존 Flexvol만 새 PV 프로비저닝에 사용됩니다.	

formatOptions 사용 권장 사항

Trident는 포맷 프로세스를 신속하게 진행하기 위해 다음 옵션을 권장합니다.

- **-E nodiscard (ext3, ext4):** mkfs 실행 시 블록을 버리지 않도록 합니다(초기 블록 버리기는 솔리드 스테이트 장치 및 스파스/씬 프로비저닝 스토리지에서 유용합니다). 이 옵션은 더 이상 사용되지 않는 "-K" 옵션을 대체하며 ext3, ext4 파일 시스템에 적용됩니다.
- **-K (xfs):** mkfs 실행 시 블록을 버리려고 시도하지 않습니다. 이 옵션은 xfs 파일 시스템에 적용됩니다.

Active Directory 자격 증명을 사용하여 **Trident**를 백엔드 **SVM**에 인증합니다

Active Directory(AD) 자격 증명을 사용하여 백엔드 SVM에 인증하도록 Trident를 구성할 수 있습니다. AD 계정이 SVM에 액세스하려면 먼저 클러스터 또는 SVM에 대한 AD 도메인 컨트롤러 액세스를 구성해야 합니다. AD 계정을 사용하여 클러스터를 관리하려면 도메인 터널을 생성해야 합니다. 자세한 내용은 "ONTAP에서 Active Directory 도메인 컨트롤러 액세스 구성"을 참조하십시오.

단계

1. 백엔드 SVM에 대한 DNS(Domain Name System) 설정을 구성합니다.

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

2. Active Directory에서 SVM용 컴퓨터 계정을 생성하려면 다음 명령을 실행하십시오.

```
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 생성합니다

```
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 `username` 및 `password` 매개 변수를 각각 AD 사용자 또는 그룹 이름과 암호로 설정하십시오.

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 `defaults` 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
`spaceAllocation`	LUN의 공간 할당	"true" 지정된 경우 **ASA r2** 시스템의 경우 `**true**`로 설정하십시오.
`spaceReserve`	공간 예약 모드: "none"(씬) 또는 "volume"(씩). **ASA r2** 시스템의 경우 `none`로 설정합니다.	"없음"
`snapshotPolicy`	사용할 스냅샷 정책입니다. **ASA r2** 시스템의 경우 `**none**`로 설정하세요.	"없음"
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유하지 않는 QoS 정책 그룹을 사용하고 각 구성 요소에 개별적으로 정책 그룹이 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 상한선을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오	""
`snapshotReserve`	스냅샷에 할당할 볼륨 비율입니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	`snapshotPolicy`이(가) "none"이면 "0", 그렇지 않으면 ""
`splitOnClone`	생성 시 상위 항목에서 클론 분할	"false"
`encryption`	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	"false" 지정된 경우 **ASA r2** 시스템의 경우 `**true**`로 설정하십시오.
`luksEncryption`	LUKS 암호화를 활성화합니다. 을 참조하십시오"Linux Unified Key Setup(LUKS) 사용".	"" **ASA r2** 시스템의 경우 `**false**`로 설정하세요.
`tieringPolicy`	계층화 정책에서 "none"을 사용합니다. **ASA r2** 시스템의 경우 지정하지 마십시오.	
`nameTemplate`	사용자 지정 볼륨 이름을 생성하기 위한 템플릿입니다.	""

볼륨 프로비저닝 예

다음은 기본값이 정의된 예입니다.

```
---
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'
```

ontap-san 드라이버를 사용하여 생성된 모든 볼륨에 대해 Trident는 LUN 메타데이터를 수용하기 위해 FlexVol에 10%의 추가 용량을 더합니다. LUN은 사용자가 PVC에서 요청한 정확한 크기로 프로비저닝됩니다. Trident는 FlexVol에 10%를 추가합니다(ONTAP에서 사용 가능한 크기로 표시됨). 이제 사용자는 요청한 만큼의 사용 가능한 용량을 확보할 수 있습니다. 또한 이 변경 사항은 사용 가능한 공간이 완전히 활용되지 않는 한 LUN이 읽기 전용이 되는 것을 방지합니다. 이는 ontap-san-economy에는 적용되지 않습니다.

`snapshotReserve`을(를) 정의하는 백엔드의 경우 Trident는 다음과 같이 볼륨 크기를 계산합니다.

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

1.1은 Trident가 LUN 메타데이터를 수용하기 위해 FlexVol에 추가하는 10%입니다. snapshotReserve = 5%이고 PVC 요청 = 5GiB인 경우 전체 볼륨 크기는 5.79GiB이고 사용 가능한 크기는 5.5GiB입니다. volume show 명령은 다음 예와 유사한 결과를 표시해야 합니다.

현재로서는 크기 조정만이 기존 볼륨에 대한 새 계산을 사용하는 유일한 방법입니다.

최소 구성 예

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

> Amazon FSx for NetApp ONTAP에서 Trident를 사용하는 경우 NetApp에서는 LIF에 IP 주소 대신 DNS 이름을 지정하는 것이 좋습니다.

ONTAP SAN 예

이는 `ontap-san` 드라이버를 사용한 기본 구성입니다.

```
---
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 예시

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

원활한 전환 및 복귀를 위해 `managementLIF`을 사용하여 `SVM`을 지정하고 `svm` 매개변수를 생략하십시오. 예를 들면 다음과 같습니다.

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

ONTAP SAN 이코노미 예

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

인증서 기반 인증 예

이 기본 구성 예에서 `clientCertificate`, `clientPrivateKey` 및 `trustedCACertificate`(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 `backend.json`에 채워지며 각각 클라이언트 인증서, 개인 키 및 신뢰할 수 있는 CA 인증서의 base64로 인코딩된 값을 사용합니다.

```
---
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** 예

이 예제들은 `useCHAP`로 설정된 `true` 백엔드를 생성합니다.

ONTAP SAN CHAP 예

```
---
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** 예

```
---
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 예

ONTAP 백엔드에 NVMe가 구성된 SVM이 있어야 합니다. NVMe/TCP를 위한 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 예

ONTAP 백엔드에 FC가 구성된 SVM이 있어야 합니다. 다음은 FC를 위한 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}}"
```

formatOptions 예시(**ontap-san-economy** 드라이버용)

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

가상 풀이 있는 백엔드의 예

이 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 특정 기본값(예: `spaceReserve`없음, `spaceAllocation`false, `encryption`false)이 설정되어 있습니다. 가상 풀은 스토리지 섹션에서 정의됩니다.

Trident는 "설명" 필드에 프로비저닝 레이블을 설정합니다. 설명은 FlexVol 볼륨에 설정됩니다. Trident는 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨으로 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이 예시에서 일부 스토리지 풀은 자체 `spaceReserve`, `spaceAllocation` 및 `encryption` 값을 설정하고, 일부 풀은 기본값을 재정의합니다.

ONTAP SAN 예

```yaml
---
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 이코노미 예

```
---
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 예

```
---
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 **StorageClasses**에 매핑합니다

다음 StorageClass 정의는 가상 풀이 있는 백엔드의 예을 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

- `protection-gold` StorageClass는 `ontap-san` 백엔드의 첫 번째 가상 풀에 매핑됩니다. 이 풀만이 골드 레벨 보호를 제공합니다.

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

- `protection-not-gold` StorageClass는 `ontap-san` 백엔드의 두 번째 및 세 번째 가상 풀에 매핑됩니다. 이 두 풀만이 골드 수준 외에 다른 보호 수준을 제공합니다.

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

- `app-mysqldb` StorageClass는 `ontap-san-economy` 백엔드의 세 번째 가상 풀에 매핑됩니다. 이 풀은 mysqldb 유형 애플리케이션에 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"
```

- `protection-silver-creditpoints-20k` StorageClass는 `ontap-san` 백엔드의 두 번째 가상 풀에 매핑됩니다. 이 풀은 실버 등급 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"
```

- `creditpoints-5k` StorageClass는 `ontap-san` 백엔드의 세 번째 가상 풀과 `ontap-san-economy` 백엔드의 네 번째 가상 풀에 매핑됩니다. 5000 크레딧 포인트를 제공하는 풀은 이 두 가지뿐입니다.

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

- `my-test-app-sc` StorageClass는 `testAPP` 드라이버의 `ontap-san` 가상 풀에 `sanType: nvme`로 매핑됩니다. 이것이 `testApp`을(를) 제공하는 유일한 풀입니다.

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

Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 보장합니다.

ONTAP NAS 드라이버

ONTAP NAS 드라이버 개요

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

ONTAP NAS 드라이버 세부 정보

Trident는 ONTAP 클러스터와의 통신을 위해 다음과 같은 NAS 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	**volumeMode**	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`

- `ontap-san-economy`지속적 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되는 경우에만 사용하십시오.
- `ontap-nas-economy`영구 볼륨 사용 수가 "지원되는 ONTAP 볼륨 제한"보다 높을 것으로 예상되고 `ontap-san-economy` 드라이버를 사용할 수 없는 경우에만 사용하십시오.
- 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우에는 `ontap-nas-economy`사용하지 마십시오.
- NetApp은 ontap-san을 제외한 모든 ONTAP 드라이버에서 Flexvol 자동 확장을 사용하지 않는 것을 권장합니다. 해결 방법으로 Trident는 스냅샷 예약 기능을 지원하며, 이에 따라 Flexvol 볼륨을 확장합니다.

사용자 권한

Trident는 일반적으로 `admin` 클러스터 사용자 또는 `vsadmin` SVM 사용자, 혹은 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상됩니다.

Amazon FSx for NetApp ONTAP 배포의 경우 Trident는 cluster `fsxadmin` 사용자 또는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름의 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행될 것으로 예상됩니다. `fsxadmin` 사용자는 클러스터 관리자 사용자를 제한적으로 대체합니다.

`limitAggregateUsage` 매개변수를 사용하는 경우 클러스터 관리자 권한이 필요합니다. Trident와 함께 Amazon FSx for NetApp ONTAP를 사용하는 경우 `limitAggregateUsage` 매개변수는 `vsadmin` 및 `fsxadmin` 사용자 계정에서 작동하지 않습니다. 이 매개변수를 지정하면 구성 작업이 실패합니다.

ONTAP 내에서 Trident 드라이버가 사용할 수 있는 더욱 제한적인 역할을 생성하는 것도 가능하지만 권장하지 않습니다. 대부분의 새로운 Trident 릴리스에서는 추가 API를 호출하므로 이를 고려해야 하기 때문에 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

ONTAP NAS 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오

ONTAP NAS 드라이버를 사용하여 ONTAP 백엔드를 구성하기 위한 요구 사항, 인증 옵션 및 엑스포트 정책을 이해하십시오.

25.10 릴리스부터 NetApp Trident는 "NetApp AFX 스토리지 시스템"을(를) 지원합니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현에서 다른 ONTAP 시스템(ASA, AFF, FAS)과 다릅니다.

AFX 시스템에서는 `ontap-nas` 드라이버(NFS 프로토콜 사용)만 지원되며 SMB 프로토콜은 지원되지 않습니다.

Trident 백엔드 구성에서 시스템이 AFX인지 여부를 지정할 필요가 없습니다. `ontap-nas`을(를) `storageDriverName`(으)로 선택하면 Trident가 AFX 시스템을 자동으로 감지합니다.

요구 사항

- 모든 ONTAP 백엔드의 경우 Trident를 사용하려면 SVM에 하나 이상의 애그리게이트를 할당해야 합니다.
- 여러 드라이버를 실행하고, 각 드라이버를 가리키는 스토리지 클래스를 생성할 수 있습니다. 예를 들어, `ontap-nas` 드라이버를 사용하는 Gold 클래스와 `ontap-nas-economy` 드라이버를 사용하는 Bronze 클래스를 구성할 수 있습니다.
- 모든 Kubernetes 워커 노드에는 적절한 NFS 도구가 설치되어 있어야 합니다. "여기"자세한 내용은 를 참조하십시오.
- Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 자세한 내용은 SMB 볼륨 프로비저닝 준비를 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 이 모드는 ONTAP 백엔드에 대한 충분한 권한이 필요합니다. ONTAP 버전과의 최대 호환성을 보장하기 위해 `admin` 또는 `vsadmin`와 같이 미리 정의된 보안 로그인 역할과 연결된 계정을 사용하는 것이 좋습니다.
- 인증서 기반: 이 모드에서는 Trident가 ONTAP 클러스터와 통신하기 위해 백엔드에 인증서가 설치되어 있어야 합니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키, 그리고 사용하는 경우(권장) 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값이 포함되어야 합니다.

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

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

자격 증명 기반 인증 활성화

Trident는 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자의 자격 증명이 필요합니다. `admin` 또는 `vsadmin`와 같은 표준 사전 정의된 역할을 사용하는 것이 좋습니다. 이렇게 하면 향후 Trident 릴리스에서 사용할 수 있는 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와의 상위 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 생성하여 Trident와 함께 사용할 수 있지만 권장하지 않습니다.

백엔드 정의 샘플은 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 평문으로 저장되는 유일한 위치라는 점을 명심하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 시크릿으로 저장됩니다. 백엔드 생성/업데이트는 자격 증명을 알아야 하는 유일한 단계입니다. 따라서 이는 Kubernetes/스토리지 관리자가 수행하는 관리자 전용 작업입니다.

인증서 기반 인증 활성화

신규 및 기존 백엔드는 인증서를 사용하여 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에는 세 가지 매개변수가 필요합니다.

- clientCertificate: 클라이언트 인증서의 Base64 인코딩 값입니다.
- clientPrivateKey: 연결된 개인 키의 Base64 인코딩 값입니다.
- trustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개변수를 반드시 제공해야 합니다. 신뢰할 수 있는 CA를 사용하지 않는 경우에는 이 매개변수를 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서와 키를 생성합니다. 생성 시 CN(일반 이름)을 인증할 ONTAP 사용자로 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. 스토리지 관리자가 이미 처리했을 수 있습니다. 신뢰할 수 있는 CA를 사용하지 않는 경우 이 단계를 건너뛰십시오.

```
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 인증서 및 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 `cert` 인증 방법을 지원하는지 확인합니다.

```
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 테스트하십시오. <ONTAP Management LIF> 및 <vserver name>를 관리 LIF IP 및 SVM 이름으로 바꾸십시오. LIF의 서비스 정책이 `default-data-management`로 설정되어 있는지 확인해야 합니다.

```
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. 인증서, 키 및 신뢰할 수 있는 CA 인증서를 Base64로 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 방법을 업데이트하거나 자격 증명을 변경하세요

기존 백엔드를 업데이트하여 다른 인증 방법을 사용하거나 자격 증명을 교체할 수 있습니다. 이 기능은 양방향으로 작동합니다. 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하는 방식으로 업데이트할 수 있고, 인증서를 사용하는 백엔드를 사용자 이름/암호 기반으로 업데이트할 수도 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개변수가 포함된 업데이트된 backend.json 파일을 사용하여 `tridentctl update backend`을(를) 실행합니다.

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

암호를 교체할 때는 스토리지 관리자가 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 후 백엔드 업데이트를 진행합니다. 인증서를 교체할 때는 사용자에게 여러 개의 인증서를 추가할 수 있습니다. 백엔드를 업데이트하여 새 인증서를 사용하도록 설정한 후, ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스는 중단되지 않으며, 이후에 생성된 볼륨 연결에도 영향을 미치지 않습니다. 백엔드 업데이트가 성공적으로 완료되면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 의미합니다.

Trident용 사용자 지정 ONTAP 역할 생성

Trident에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 권한으로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident는 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 "Trident 사용자 지정 역할 생성기"을 참조하십시오.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 생성합니다.

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

2. Trident 사용자의 사용자 이름을 생성합니다.

```
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 역할 매핑:

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

System Manager 사용

ONTAP System Manager에서 다음 단계를 수행하십시오.

1. 사용자 지정 역할 생성:
   a. 클러스터 수준에서 사용자 지정 역할을 생성하려면 * Cluster > Settings * 를 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 생성하려면 *스토리지 > 스토리지 VM > `required SVM`> 설정 > 사용자 및 역할*을 선택하십시오.

   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 아래에서 추가 아이콘 *+*를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

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

NFS 엑스포트 정책 관리

Trident는 NFS 엑스포트 정책을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어합니다.

Trident는 엑스포트 정책 작업 시 두 가지 옵션을 제공합니다.

- Trident는 엑스포트 정책을 동적으로 관리할 수 있습니다. 이 운영 모드에서 스토리지 관리자는 허용 가능한 IP 주소를 나타내는 CIDR 블록 목록을 지정합니다. Trident는 게시 시점에 이러한 범위에 속하는 해당 노드 IP를 엑스포트 정책에 자동으로 추가합니다. 또는 CIDR이 지정되지 않은 경우, 볼륨이 게시될 노드에서 발견된 모든 글로벌 범위 유니캐스트 IP가 엑스포트 정책에 추가됩니다.
- 스토리지 관리자는 엑스포트 정책을 생성하고 규칙을 수동으로 추가할 수 있습니다. Trident는 구성 파일에 다른 엑스포트 정책 이름이 지정되지 않은 경우 기본 엑스포트 정책을 사용합니다.

엑스포트 정책을 동적으로 관리합니다

Trident는 ONTAP 백엔드에 대한 엑스포트 규칙을 동적으로 관리할 수 있는 기능을 제공합니다. 이를 통해 스토리지 관리자는 명시적인 규칙을 수동으로 정의하는 대신 워커 노드 IP에 대해 허용 가능한 주소 공간을 지정할 수 있습니다. 이는 엑스포트 규칙 관리를 크게 간소화하며, 엑스포트 규칙을 수정할 때 더 이상 스토리지 클러스터에서 수동으로 개입할 필요가 없습니다. 또한, 이 기능을 통해 볼륨을 마운트하고 지정된 IP 범위 내에 있는 워커 노드만 스토리지 클러스터에 액세스할 수 있도록 제한하여 세밀하고 자동화된 관리를 지원합니다.

> 동적 엑스포트 규칙을 사용할 때는 네트워크 주소 변환(NAT)을 사용하지 마십시오. NAT를 사용하면 스토리지 컨트롤러가 실제 IP 주소가 아닌 프런트엔드 NAT 주소를 인식하게 되므로 엑스포트 규칙에서 일치하는 항목이 없으면 액세스가 거부됩니다.

예

반드시 사용해야 하는 구성 옵션이 두 가지 있습니다. 다음은 백엔드 정의 예시입니다.

```
---
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 사용할 때는 SVM의 루트 접합부에 노드 CIDR 블록을 허용하는 엑스포트 규칙(예: 기본 엑스포트 정책)이 포함된 엑스포트 정책이 미리 생성되어 있는지 확인해야 합니다. 항상 NetApp 권장 모범 사례에 따라 Trident 전용 SVM을 사용하십시오.

다음은 위의 예를 사용하여 이 기능이 작동하는 방식에 대한 설명입니다.

- `autoExportPolicy`이(가) `true`(으)로 설정되어 있습니다. 이는 Trident가 `svm1` SVM에 대해 이 백엔드로 프로비저닝된 각 볼륨에 대한 엑스포트 정책을 생성하고 `autoexportCIDRs` 주소 블록을 사용하여 규칙 추가 및 삭제를 처리함을 나타냅니다. 볼륨이 노드에 연결될 때까지 해당 볼륨은 원치 않는 액세스를 방지하기 위해 규칙이 없는 빈 엑스포트 정책을 사용합니다. 볼륨이 노드에 게시되면 Trident는 지정된 CIDR 블록 내의 노드 IP를 포함하는 기본 qtree와 동일한 이름의 엑스포트 정책을 생성합니다. 이러한 IP는 상위 FlexVol 볼륨에서 사용하는 엑스포트 정책에도 추가됩니다.

◦ 예를 들면 다음과 같습니다.

- 백엔드 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
- `autoExportPolicy` 로 설정 `true`
- 스토리지 접두사 `trident`
- PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
- trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c라는 이름의 qtree는 FlexVol에 대한 엑스포트 규칙 `trident-403b5326-8482-40db96d0-d83fb3f4daec`, qtree에 대한 엑스포트 규칙 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, 그리고 SVM에 빈 엑스포트 규칙 `trident_empty`을 생성합니다. FlexVol 엑스포트 규칙의 규칙은 qtree 엑스포트 규칙에 포함된 모든 규칙의 상위 집합입니다. 빈 엑스포트 규칙은 연결되지 않은 볼륨에서 재사용됩니다.

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

이 예에서는 `192.168.0.0/24` 주소 공간이 제공됩니다. 이는 게시와 함께 이 주소 범위 내에 속하는 Kubernetes 노드 IP가 Trident가 생성하는 엑스포트 정책에 추가됨을 나타냅니다. Trident가 실행 중인 노드를 등록할 때 해당 노드의 IP 주소를 검색하고 `autoExportCIDRs`에 제공된 주소 블록과 비교합니다. 게시 시점에 IP를 필터링한 후 Trident는 게시 대상 노드의 클라이언트 IP에 대한 엑스포트 정책 규칙을 생성합니다.

> `autoExportPolicy`및 `autoExportCIDRs`를 생성한 후 백엔드에 대해 업데이트할 수 있습니다. 자동으로 관리되는 백엔드에 대해 새 `CIDR`을 추가하거나 기존 `CIDR`을 삭제할 수 있습니다. 기존 연결이 끊어지지 않도록 `CIDR`을 삭제할 때 주의하십시오. 또한 백엔드에 대해 `autoExportPolicy`를 비활성화하고 수동으로 생성된 엑스포트 정책으로 대체하도록 선택할 수 있습니다. 이렇게 하려면 백엔드 구성에서 `exportPolicy` 매개 변수를 설정해야 합니다.

Trident가 백엔드를 생성하거나 업데이트한 후 `tridentctl` 또는 해당 `tridentbackend` CRD를 사용하여 백엔드를 확인할 수 있습니다.

```
./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 Trident는 모든 엑스포트 정책을 확인하여 해당 노드에 해당하는 액세스 규칙을 제거합니다. 관리형 백엔드의 엑스포트 정책에서 이 노드 IP를 제거함으로써 Trident는 클러스터의 새 노드에서 이 IP가 재사용되지 않는 한 무단 마운트를 방지합니다.

기존 백엔드의 경우 `tridentctl update backend`로 백엔드를 업데이트하면 Trident에서 엑스포트 정책을 자동으로 관리합니다. 이렇게 하면 필요할 때 백엔드의 UUID와 qtree 이름을 따서 명명된 두 개의 새 엑스포트 정책이 생성됩니다. 백엔드에 있는 볼륨은 마운트 해제 후 다시 마운트되면 새로 생성된 엑스포트 정책을 사용합니다.

> 자동 관리되는 엑스포트 규칙이 있는 백엔드를 삭제하면 동적으로 생성된 엑스포트 규칙도 삭제됩니다. 백엔드를 다시 생성하면 새 백엔드로 간주되어 새 엑스포트 규칙이 생성됩니다.

실행 중인 노드의 IP 주소가 업데이트되면 해당 노드에서 Trident Pod를 재시작해야 합니다. 그러면 Trident는 관리하는 백엔드에 대한 엑스포트 정책을 업데이트하여 이 IP 변경 사항을 반영합니다.

SMB 볼륨 프로비저닝 준비

약간의 추가 준비 작업을 거치면 `ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다.

> ONTAP 온프레미스 클러스터용 `ontap-nas-economy` SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 합니다. 이러한 프로토콜 중 하나라도 구성하지 않으면 SMB 볼륨 생성이 실패합니다.

> `autoExportPolicy`는 SMB 볼륨에서 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하기 전에 다음 사항을 충족해야 합니다.

- Linux 컨트롤러 노드와 Windows Server 2022를 실행하는 하나 이상의 Windows 워커 노드로 구성된 Kubernetes 클러스터입니다. Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 비밀 키가 하나 이상 필요합니다. 비밀 키를 생성하려면 `smbcreds`:

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

- Windows 서비스로 구성된 CSI 프록시입니다. `csi-proxy`를 구성하려면 Windows에서 실행되는 Kubernetes 노드에 대한 "GitHub: CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시"를 참조하십시오.

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident가 대신 생성해 줄 수 있습니다.

   Amazon FSx for ONTAP에는 SMB 공유가 필요합니다.

   SMB 관리자 공유는 "Microsoft Management Console" 공유 폴더 스냅인을 사용하거나 ONTAP CLI를 사용하는 두 가지 방법으로 생성할 수 있습니다. ONTAP CLI를 사용하여 SMB 공유를 생성하려면 다음을 수행합니다.

   a. 필요한 경우 공유에 대한 디렉터리 경로 구조를 생성하십시오.

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

   b. 지정된 SVM과 연결된 SMB 공유를 생성합니다.

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

   c. 공유가 생성되었는지 확인합니다.

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

      자세한 내용은 "SMB 공유 생성"를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 FSx for ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 "FSx for ONTAP 구성 옵션 및 예"를 참조하십시오.

매개변수	설명	예
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	`smb-share`
`nasType`	* `smb`로 설정해야 합니다.* null인 경우 기본값은 `nfs`입니다.	`smb`
`securityStyle`	새 볼륨에 대한 보안 스타일입니다. **SMB** 볼륨의 경우 **`ntfs`** 또는 **`mixed`**로 설정해야 합니다**.**	`ntfs` 또는 `mixed` SMB 볼륨의 경우
`unixPermissions`	새 볼륨의 모드입니다. **SMB** 볼륨의 경우 비워 두어야 합니다**.**	""

보안 **SMB** 활성화

25.06 릴리스부터 NetApp Trident는 `ontap-nas` 및 `ontap-nas-economy` 백엔드를 사용하여 생성된 SMB 볼륨의 보안 프로비저닝을 지원합니다. 보안 SMB가 활성화되면 액세스 제어 목록(ACL)을 사용하여 Active Directory(AD) 사용자 및 사용자 그룹에 SMB 공유에 대한 제어된 액세스를 제공할 수 있습니다.

기억해야 할 사항

- `ontap-nas-economy` 볼륨 가져오기는 지원되지 않습니다.
- `ontap-nas-economy` 볼륨의 경우 읽기 전용 클론만 지원됩니다.
- Secure SMB가 활성화된 경우 Trident는 백엔드에 언급된 SMB 공유를 무시합니다.
- PVC 주석, 스토리지 클래스 주석 및 백엔드 필드를 업데이트해도 SMB 공유 ACL은 업데이트되지 않습니다.
- 클론 PVC의 주석에 지정된 SMB 공유 ACL은 소스 PVC의 ACL보다 우선합니다.
- 보안 SMB를 활성화할 때 유효한 AD 사용자를 제공해야 합니다. 유효하지 않은 사용자는 ACL에 추가되지 않습니다.
- 백엔드, 스토리지 클래스 및 PVC에 동일한 AD 사용자를 지정하고 각기 다른 권한을 부여하는 경우 권한 우선순위는 PVC, 스토리지 클래스, 백엔드 순이 됩니다.
- 보안 SMB는 `ontap-nas` 관리형 볼륨 가져오기에 대해 지원되며, 관리되지 않는 볼륨 가져오기에는 적용되지 않습니다.

단계

1. 다음 예시와 같이 TridentBackendConfig에서 adAdminUser를 지정하십시오:

```
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. 스토리지 클래스에 주석을 추가합니다.

`trident.netapp.io/smbShareAdUser` 어노테이션을 스토리지 클래스에 추가하여 안전한 SMB를 확실하게 활성화하십시오. 어노테이션 `trident.netapp.io/smbShareAdUser`에 지정된 사용자 값은 `smbcreds` 시크릿에 지정된 사용자 이름과 동일해야 합니다. `smbShareAdUserPermission`에 대해 다음 중 하나를 선택할 수 있습니다: `full_control`, `change` 또는 `read`. 기본 권한은 `full_control`입니다.

```
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 생성합니다.

다음 예제는 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 설치 환경에서 ONTAP NAS 드라이버를 생성하고 사용하는 방법을 알아보세요. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다.

25.10 릴리스부터 NetApp Trident는 "NetApp AFX 스토리지 시스템"를 지원합니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현에서 다른 ONTAP 기반 시스템(ASA, AFF, FAS)과 다릅니다.

> ⓘ `ontap-nas` 드라이버(NFS 프로토콜 사용)만 NetApp AFX 시스템에서 지원되며 SMB 프로토콜은 지원되지 않습니다.

Trident 백엔드 구성에서 시스템이 NetApp AFX 스토리지 시스템임을 지정할 필요가 없습니다. `ontap-nas`을 `storageDriverName`로 선택하면 Trident가 AFX 스토리지 시스템을 자동으로 감지합니다. 아래 표에 나와 있는 것처럼 일부 백엔드 구성 매개변수는 AFX 스토리지 시스템에 적용되지 않습니다.

백엔드 **configuration** 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값
`version`		항상 1

매개변수	설명	기본값
`storageDriverName`	스토리지 드라이버의 이름 NetApp AFX 시스템의 경우 `ontap-nas`만 지원됩니다.	`ontap-nas`, `ontap-nas-economy` 또는 `ontap-nas-flexgroup`
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호로 정의해야 합니다. 원활한 MetroCluster 전환에 대해서는 MetroCluster 예시를 참조하십시오.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	프로토콜 LIF의 IP 주소입니다. NetApp에서는 `dataLIF`을 지정하는 것을 권장합니다. 지정하지 않으면 Trident가 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 생성할 수 있습니다. 초기 설정 후 변경할 수 있습니다. 을 참조하십시오. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`와 같이 대괄호 안에 정의해야 합니다. **Metrocluster**의 경우 생략합니다**.** MetroCluster 예시을 참조하십시오.	지정된 주소 또는 지정되지 않은 경우 SVM에서 파생(권장하지 않음)
`svm`	사용할 스토리지 가상 머신 **MetroCluster**의 경우 생략 MetroCluster 예시을 참조하십시오.	SVM `managementLIF`이 지정된 경우 파생됩니다
`autoExportPolicy`	자동 엑스포트 정책 생성 및 업데이트를 활성화합니다[Boolean]. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident에서 엑스포트 정책을 자동으로 관리할 수 있습니다.	거짓
`autoExportCIDRs`	`autoExportPolicy`이 활성화된 경우 Kubernetes 노드 IP를 필터링하는 데 사용할 CIDR 목록입니다. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	["0.0.0.0/0", "::/0"]`
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
`clientPrivateKey`	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다.	""
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다	""

매개변수	설명	기본값
`username`	클러스터/SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대한 자세한 내용은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	
`password`	클러스터/SVM에 연결하는 데 사용되는 암호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대한 자세한 내용은 "Active Directory 자격 증명을 사용하여 Trident를 백엔드 SVM에 인증합니다"을(를) 참조하십시오.	
`storagePrefix`	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 설정 후에는 업데이트할 수 없습니다 ontap-nas-economy를 사용하고 storagePrefix가 24자 이상인 경우, qtree에는 스토리지 접두사가 포함되지 않지만 볼륨 이름에는 포함됩니다.	"Trident"
`aggregate`	프로비저닝용 애그리게이트(선택 사항, 설정된 경우 SVM에 할당해야 함). `ontap-nas-flexgroup` 드라이버의 경우 이 옵션은 무시됩니다. 할당하지 않으면 사용 가능한 애그리게이트 중 하나를 사용하여 FlexGroup 볼륨을 프로비저닝할 수 있습니다. SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident에서 백엔드가 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **AFX** 스토리지 시스템에는 지정하지 마십시오.	""
`limitAggregateUsage`	사용량이 이 비율을 초과하면 프로비저닝을 실패시킵니다. **Amazon FSx for ONTAP**에는 적용되지 않습니다. **AFX** 스토리지 시스템에는 지정하지 마십시오.	"" (기본적으로 적용되지 않음)

매개변수	설명	기본값
flexgroupAggregateList	프로비저닝을 위한 애그리게이트 목록(선택 사항, 설정된 경우 SVM에 할당되어야 함). SVM에 할당된 모든 애그리게이트는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다. **ontap-nas-flexgroup** 스토리지 드라이버에서 지원됩니다. ⓘ SVM에서 애그리게이트 목록이 업데이트되면 Trident Controller를 재시작하지 않고도 SVM을 폴링하여 Trident의 목록이 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트 목록을 구성한 경우, 해당 애그리게이트 목록의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 Trident의 백엔드 상태가 실패로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 애그리게이트 목록을 SVM에 있는 다른 목록으로 변경하거나 완전히 제거해야 합니다.	""
`limitVolumeSize`	요청된 볼륨 크기가 이 값보다 크면 프로비저닝이 실패합니다.	"" (기본적으로 적용되지 않음)
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 `debugTraceFlags` 사용하지 마십시오.	null
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 `nfs`, `smb` 또는 null입니다. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다. **AFX** 스토리지 시스템의 경우 지정된 경우 항상 `**nfs**`로 설정해야 합니다.	`nfs`
`nfsMountOptions`	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident는 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
`qtreesPerFlexvol`	FlexVol당 최대 Qtree 수는 [50, 300] 범위 내에 있어야 합니다	"200"
`smbShare`	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름, Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름, 또는 볼륨에 대한 공통 공유 액세스를 방지하려면 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 온프레미스 ONTAP의 경우 선택 사항입니다. 이 매개 변수는 Amazon FSx for ONTAP 백엔드에 필요하며 비워 둘 수 없습니다.	`smb-share`

매개변수	설명	기본값
useREST	ONTAP REST API를 사용하는 부울 매개 변수입니다. useREST`로 설정하면 `true Trident는 ONTAP REST API를 사용하여 백엔드와 통신하고, false`로 설정하면 Trident는 ONTAPI(ZAPI) 호출을 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontapi 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 vsadmin 및 cluster-admin 역할로 충족됩니다. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST`는 기본적으로 `true`로 설정되며, ONTAPI(ZAPI) 호출을 사용하려면 `useREST`를 `false`로 변경하십시오. **AFX** 스토리지 시스템의 경우 지정된 경우 항상 **`true`**로 설정하십시오.	ONTAP 9.15.1 이상의 경우 true, 그렇지 않은 경우 false.
limitVolumePoolSize	ontap-nas-economy 백엔드에서 Qtree를 사용할 때 요청할 수 있는 최대 FlexVol 크기입니다.	"" (기본적으로 적용되지 않음)
denyNewVolumePools	`ontap-nas-economy`백엔드가 Qtree를 포함할 새 FlexVol 볼륨을 생성하지 못하도록 제한합니다. 기존 Flexvol만 새 PV 프로비저닝에 사용됩니다.	
adAdminUser	SMB 공유에 대한 전체 액세스 권한이 있는 Active Directory 관리자 사용자 또는 사용자 그룹입니다. 이 매개변수를 사용하여 SMB 공유에 대한 전체 제어 권한이 있는 관리자 권한을 제공합니다.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 defaults 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
spaceAllocation	Qtree에 대한 공간 할당	"true"
spaceReserve	공간 예약 모드, "none"(씬) 또는 "volume"(씩)	"없음"
snapshotPolicy	사용할 스냅샷 정책	"없음"
qosPolicy	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다	""
adaptiveQosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. ontap-nas-economy에서는 지원되지 않습니다.	""
snapshotReserve	스냅샷용으로 예약된 볼륨의 비율	`snapshotPolicy`이(가) "none"이면 "0", 그렇지 않으면 ""
splitOnClone	생성 시 상위 항목에서 클론 분할	"false"

매개변수	설명	기본값
`encryption`	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	"false"
`tieringPolicy`	계층화 정책에서 "none"을 사용합니다.	
`unixPermissions`	새 볼륨용 모드	NFS 볼륨의 경우 "777", SMB 볼륨의 경우 비어 있음(해당 없음)
`snapshotDir`	`.snapshot` 디렉터리에 대한 액세스를 제어합니다	NFSv4의 경우 "true", NFSv3의 경우 "false"
`exportPolicy`	사용할 엑스포트 정책	"default"
`securityStyle`	새 볼륨에 대한 보안 스타일. NFS는 `mixed` 및 `unix` 보안 스타일을 지원합니다. SMB는 `mixed` 및 `ntfs` 보안 스타일을 지원합니다.	NFS 기본값은 `unix`입니다. SMB 기본값은 `ntfs`입니다.
`nameTemplate`	사용자 지정 볼륨 이름을 생성하기 위한 템플릿입니다.	""

Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유하지 않는 QoS 정책 그룹을 사용하고 각 구성 요소에 개별적으로 정책 그룹을 적용해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 상한선을 적용합니다.

볼륨 프로비저닝 예

다음은 기본값이 정의된 예입니다.

```
---
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"
```

`ontap-nas` 및 `ontap-nas-flexgroups`의 경우, Trident는 이제 FlexVol이 snapshotReserve 비율과 PVC에 맞게 올바르게 크기가 지정되도록 새로운 계산 방식을 사용합니다. 사용자가 PVC를 요청하면, Trident는 새로운 계산을 사용하여 더 많은 공간을 가진 원래 FlexVol을 생성합니다. 이 계산은 사용자가 PVC에서 요청한 만큼의 쓰기 가능한 공간을 받을 수 있도록 하며, 요청한 것보다 적은 공간을 제공하지 않습니다. v21.07 이전에는 사용자가 PVC(예: 5GiB)를 요청하고 snapshotReserve를 50퍼센트로 설정하면, 쓰기 가능한 공간은 2.5GiB만 제공되었습니다. 이는 사용자가 요청한 것이 전체 볼륨이고 `snapshotReserve`가 그 전체의 비율이기 때문입니다. Trident 21.07부터는 사용자가 요청하는 것이 쓰기 가능한 공간이며, Trident는 `snapshotReserve` 숫자를 전체 볼륨의 비율로 정의합니다. 이 내용은 `ontap-nas-economy`에는 적용되지 않습니다. 다음 예제를 통해 이 동작 방식을 확인할 수 있습니다:

계산 방법은 다음과 같습니다.

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

snapshotReserve = 50%이고 PVC 요청 = 5GiB인 경우, 전체 볼륨 크기는 5/.5 = 10GiB이고 사용 가능한 크기는

5GiB로, 사용자가 PVC 요청에서 요청한 크기입니다. `volume show` 명령은 다음 예시와 유사한 결과를 표시합니다.

이전 설치의 기존 백엔드는 Trident를 업그레이드할 때 위에서 설명한 대로 볼륨을 프로비저닝합니다. 업그레이드 전에 생성한 볼륨의 경우 변경 사항이 적용되도록 크기를 조정해야 합니다. 예를 들어, `snapshotReserve=50`이전에 2GiB PVC로 설정된 볼륨은 1GiB의 쓰기 공간을 제공합니다. 예를 들어 볼륨 크기를 3GiB로 조정하면 애플리케이션은 6GiB 볼륨에서 3GiB의 쓰기 공간을 사용할 수 있습니다.

최소 구성 예

다음 예시들은 대부분의 매개변수를 기본값으로 유지하는 기본 구성을 보여줍니다. 이것이 백엔드를 정의하는 가장 쉬운 방법입니다.

> Amazon FSx on NetApp ONTAP에서 Trident를 사용하는 경우 LIF에 대해 IP 주소 대신 DNS 이름을 지정하는 것이 좋습니다.

ONTAP NAS 이코노미 예

```
---
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 예

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

MetroCluster 예시

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

원활한 전환 및 복귀를 위해 `managementLIF`을 사용하여 `SVM`을 지정하고 `dataLIF` 및 `svm` 매개 변수를 생략하십시오. 예를 들면 다음과 같습니다.

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

SMB 볼륨 예

```
---
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 예

이는 최소한의 백엔드 구성 예시입니다. `clientCertificate`, `clientPrivateKey` 및 `trustedCACertificate`(신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 `backend.json`에 입력되며 클라이언트 인증서, 개인 키 및 신뢰할 수 있는 CA 인증서의 base64 인코딩된 값을 각각 사용합니다.

```
---
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 엑스포트 정책 예

이 예제는 Trident가 동적 내보내기 정책을 사용하여 내보내기 정책을 자동으로 생성하고 관리하도록 지시하는 방법을 보여줍니다. 이는 `ontap-nas-economy` 및 `ontap-nas-flexgroup` 드라이버에 대해 동일하게 작동합니다.

```
---
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 예시는 IPv6 주소를 사용하는 `managementLIF` 방법을 보여줍니다.

```
---
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 예

`smbShare` 매개 변수는 SMB 볼륨을 사용하는 FSx for ONTAP에 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 풀이 있는 백엔드의 예

아래에 표시된 샘플 백엔드 정의 파일에서는 모든 스토리지 풀에 대해 `spaceReserve`없음, `spaceAllocation`false, `encryption`false와 같은 특정 기본값이 설정되어 있습니다. 가상 풀은 스토리지 섹션에서 정의됩니다.

Trident는 "설명" 필드에 프로비저닝 레이블을 설정합니다. 설명은 FlexVol의 경우 `ontap-nas` 또는 FlexGroup의 경우 `ontap-nas-flexgroup`에 설정됩니다. Trident는 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨으로 복사합니다. 편의를 위해 스토리지 관리자는 가상 풀별로 레이블을 정의하고 레이블별로 볼륨을 그룹화할 수 있습니다.

이 예시에서 일부 스토리지 풀은 자체 `spaceReserve`, `spaceAllocation` 및 `encryption` 값을 설정하고, 일부 풀은 기본값을 재정의합니다.

ONTAP NAS 예

```
---
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 예

```
---
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 이코노미 예

```
---
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 StorageClasses에 매핑합니다

다음 StorageClass 정의는 가상 풀이 있는 백엔드의 예를 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

- `protection-gold` StorageClass는 `ontap-nas-flexgroup` 백엔드의 첫 번째 및 두 번째 가상 풀에 매핑됩니다. 이 두 풀만 골드 레벨 보호를 제공합니다.

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

- `protection-not-gold` StorageClass는 `ontap-nas-flexgroup` 백엔드의 세 번째 및 네 번째 가상 풀에 매핑됩니다. 이 두 풀만이 골드 등급 이외의 보호 수준을 제공합니다.

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

- `app-mysqldb` StorageClass는 `ontap-nas` 백엔드의 네 번째 가상 풀에 매핑됩니다. 이 풀은 mysqldb 유형 애플리케이션에 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"
```

- `protection-silver-creditpoints-20k` StorageClass는 `ontap-nas-flexgroup` 백엔드의 세 번째 가상 풀에 매핑됩니다. 이 풀은 실버 등급 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"
```

- `creditpoints-5k` StorageClass는 `ontap-nas` 백엔드의 세 번째 가상 풀과 `ontap-nas-economy` 백엔드의 두 번째 가상 풀에 매핑됩니다. 5000 크레딧 포인트를 제공하는 풀은 이 두 가지뿐입니다.

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

Trident는 어떤 가상 풀이 선택될지 결정하고 스토리지 요구사항이 충족되도록 보장합니다.

초기 구성 후 업데이트 `dataLIF`

초기 구성 후 다음 명령을 실행하여 업데이트된 dataLIF가 포함된 새 백엔드 JSON 파일을 제공함으로써 dataLIF를 변경할 수 있습니다.

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

PVC가 하나 이상의 Pod에 연결된 경우, 새 dataLIF가 적용되려면 해당 Pod를 모두 종료한 다음 다시 시작해야 합니다.

보안 **SMB** 예

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
```

`annotations`을(를) 추가하여 보안 SMB를 활성화해야 합니다. 백엔드 또는 PVC에 설정된 구성과 관계없이 어노테이션이 없으면 보안 SMB가 작동하지 않습니다.

ontap-nas-economy 드라이버를 사용한 스토리지 클래스 예

```
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** 사용자를 사용한 **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** 사용자를 포함하는 **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 ONTAP와 함께 **Trident** 사용

"Amazon FSx for NetApp ONTAP"는 NetApp ONTAP 스토리지 운영 체제를 기반으로 하는 파일 시스템을 시작하고 실행할 수 있도록 지원하는 완전 관리형 AWS 서비스입니다. FSx for ONTAP을 사용하면 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하는 동시에 AWS에 데이터를 저장하는 간편성, 민첩성, 보안 및 확장성의 이점을 누릴 수 있습니다. FSx for ONTAP은 ONTAP 파일 시스템 기능 및 관리 API를 지원합니다.

Amazon FSx for NetApp ONTAP 파일 시스템을 Trident와 통합하여 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터가 ONTAP에서 지원하는 블록 및 파일 영구 볼륨을 프로비저닝할 수 있도록 할 수 있습니다.

파일 시스템은 Amazon FSx의 기본 리소스로, 온프레미스 ONTAP 클러스터와 유사합니다. 각 SVM 내에서 하나 이상의 볼륨을 생성할 수 있으며, 볼륨은 파일 시스템의 파일과 폴더를 저장하는 데이터 컨테이너입니다. Amazon FSx for NetApp ONTAP은 클라우드에서 관리형 파일 시스템으로 제공됩니다. 이 새로운 파일 시스템 유형은 *NetApp ONTAP*이라고 합니다.

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

요구 사항

"Trident 요구 사항" 외에도 FSx for ONTAP을 Trident와 통합하려면 다음이 필요합니다.

- `kubectl`가 설치된 기존 Amazon EKS 클러스터 또는 자체 관리형 Kubernetes 클러스터.
- 클러스터의 워커 노드에서 연결할 수 있는 기존 Amazon FSx for NetApp ONTAP 파일 시스템 및 스토리지 가상 머신(SVM).
- "NFS 또는 iSCSI"에 대해 준비된 작업자 노드.

> EKS AMI 유형에 따라 Amazon Linux 및 Ubuntu "Amazon Machine Images"(AMI)에 필요한 노드 준비 단계를 반드시 따르십시오.

고려 사항

- SMB 볼륨:
  - SMB 볼륨은 `ontap-nas` 드라이버를 통해서만 지원됩니다.
  - SMB 볼륨은 Trident EKS 애드온에서 지원되지 않습니다.
  - Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 자세한 내용은 "SMB 볼륨 프로비저닝 준비"를 참조하십시오.
- Trident 24.02 이전 버전에서는 자동 백업이 활성화된 Amazon FSx 파일 시스템에 생성된 볼륨을 Trident에서 삭제할 수 없었습니다. Trident 24.02 이상 버전에서 이 문제를 방지하려면 Amazon FSx for NetApp ONTAP의 백엔드 구성 파일에서 `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey`, AWS `secretKey`를 지정하십시오.

> Trident에 IAM 역할을 지정하는 경우 `apiRegion`, `apiKey` 및 `secretKey` 필드를 Trident에 명시적으로 지정하지 않아도 됩니다. 자세한 내용은"FSx for ONTAP 구성 옵션 및 예"을 참조하십시오.

Trident SAN/iSCSI 및 **EBS-CSI** 드라이버의 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 노드의 EBS 디스크와 충돌 없이 다중 경로가 작동하도록 하려면 다중 경로 설정에서 EBS를 제외해야 합니다. 다음 예는 다중 경로에서 EBS 디스크를 제외하면서 필요한 Trident 설정을 포함하는 `multipath.conf` 파일입니다.

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

인증

Trident는 두 가지 인증 방식을 제공합니다.

- 자격 증명 기반(권장): 자격 증명을 AWS Secrets Manager에 안전하게 저장합니다. `fsxadmin` 사용자를 파일 시스템에 사용하거나 `vsadmin` 사용자를 SVM에 구성하여 사용할 수 있습니다.

> Trident는 `vsadmin` SVM 사용자 또는 동일한 역할을 가진 다른 이름의 사용자로 실행되어야 합니다. Amazon FSx for NetApp ONTAP에는 ONTAP `admin` 클러스터 사용자의 제한된 대체 역할을 하는 `fsxadmin` 사용자가 있습니다. Trident와 함께 `vsadmin` 사용을 강력히 권장합니다.

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

인증 활성화에 대한 자세한 내용은 드라이버 유형에 대한 인증을 참조하십시오.

- "ONTAP NAS 인증"
- "ONTAP SAN 인증"

테스트된 **Amazon Machine Images(AMI)**

EKS 클러스터는 다양한 운영 체제를 지원하지만, AWS는 컨테이너 및 EKS에 최적화된 특정 Amazon Machine Image(AMI)를 제공합니다. 다음 AMI는 NetApp Trident 25.02에서 테스트되었습니다.

AMI	NAS	NAS-이코노미	iSCSI	iSCSI-경제성
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 호환되지 않습니다.

> 원하는 AMI가 이 목록에 없다고 해서 지원되지 않는다는 의미는 아닙니다. 단지 테스트가 완료되지 않았다는 뜻입니다. 이 목록은 정상 작동하는 것으로 확인된 AMI를 안내하기 위한 것입니다.

다음을 사용하여 수행된 테스트**:**

- EKS 버전: 1.32
- 설치 방법: Helm 25.06 및 AWS 추가 기능 25.06
- NAS의 경우 NFSv3과 NFSv4.1이 모두 테스트되었습니다.
- SAN의 경우 iSCSI만 테스트되었으며 NVMe-oF는 테스트되지 않았습니다.

수행된 테스트:

- 생성: Storage Class, pvc, pod
- 삭제: pod, pvc(일반, qtree/lun – economy, AWS 백업이 있는 NAS)

자세한 정보 찾기

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

IAM 역할 및 **AWS Secret**을 생성합니다

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

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

AWS Secrets Manager 시크릿 생성

Trident는 스토리지 관리를 위해 FSx vserver에 대한 API를 발행하므로 자격 증명이 필요합니다. 이러한 자격 증명을 안전하게 전달하는 방법은 AWS Secrets Manager 시크릿을 사용하는 것입니다. 따라서 아직 AWS Secrets Manager 시크릿이 없다면 vsadmin 계정의 자격 증명이 포함된 시크릿을 생성해야 합니다.

이 예제는 Trident CSI 자격 증명을 저장하기 위한 AWS Secrets Manager 시크릿을 생성합니다.

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

IAM 정책 생성

Trident가 제대로 실행되려면 AWS 권한도 필요합니다. 따라서 Trident에 필요한 권한을 부여하는 정책을 생성해야 합니다.

다음 예제는 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** 예:

```
{
  "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 Identity** 또는 **IAM** 역할을 생성합니다.

Kubernetes 서비스 계정이 EKS Pod Identity 또는 IRSA(서비스 계정 연결용 IAM 역할)를 사용하는 AWS Identity and Access Management(IAM) 역할을 수임하도록 구성할 수 있습니다. 해당 서비스 계정을 사용하도록 구성된 모든 Pod는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

Pod Identity

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 사용하거나 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 "Amazon EKS Pod Identity Agent를 설정합니다"을(를) 참조하십시오.

trust-relationship.json 생성:

EKS 서비스 주체가 Pod Identity에 대한 이 역할을 수임할 수 있도록 trust-relationship.json을 생성합니다. 그런 다음 이 신뢰 정책을 사용하여 역할을 생성합니다.

```
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 연결:

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 생성합니다.

```
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 연결을 위한 **IAM** 역할**(IRSA)**

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"
        }
      }
    }
  ]
}
```

`trust-relationship.json` 파일에서 다음 값을 업데이트합니다.

- **<account_id>** - AWS 계정 ID
- **<oidc_provider>** - EKS 클러스터의 OIDC입니다. 다음 명령을 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** 역할을 연결합니다:

역할이 생성되면 이 명령을 사용하여 (위 단계에서 생성한) 정책을 역할에 연결합니다.

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

OICD 공급자가 연결되어 있는지 확인:

OIDC 공급자가 클러스터에 연결되어 있는지 확인하십시오. 다음 명령을 사용하여 확인할 수 있습니다.

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

출력이 비어 있는 경우 다음 명령을 사용하여 IAM OIDC를 클러스터에 연결하십시오.

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

eksctl을 사용하는 경우 다음 예제를 사용하여 EKS에서 서비스 계정에 대한 IAM 역할을 생성하십시오.

```
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

Trident는 Amazon FSx for NetApp ONTAP 스토리지 관리를 Kubernetes에서 간소화하여

개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다.

다음 방법 중 하나를 사용하여 Trident를 설치할 수 있습니다.

- Helm
- EKS 추가 기능

스냅샷 기능을 사용하려면 CSI 스냅샷 컨트롤러 추가 기능을 설치하십시오. 자세한 내용은 "CSI 볼륨에 대한 스냅샷 기능 활성화"를 참조하십시오.

Helm을 통해 **Trident** 설치

Pod Identity

1. Trident Helm 리포지토리 추가:

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

2. 다음 예시를 사용하여 Trident를 설치합니다.

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

`helm list` 명령을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 개정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

```
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 리포지토리 추가:

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

2. 클라우드 공급자 및 클라우드 **ID** 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
```

`helm list` 명령을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 개정 번호와 같은 설치 세부 정보를 검토할 수 있습니다.

```
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-
100.2510.0    25.10.0
```

iSCSI를 사용하려는 경우 클라이언트 시스템에서 iSCSI가 활성화되어 있는지 확인하십시오. AL2023 Worker 노드 OS를 사용하는 경우 helm 설치에 node prep 매개 변수를 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**를 설치하십시오

Trident EKS 애드온에는 최신 보안 패치와 버그 수정 사항이 포함되어 있으며, AWS에서 Amazon EKS와의 호환성을 검증받았습니다. EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 유지할 수 있으며, 애드온 설치, 구성 및 업데이트에 필요한 작업량을 줄일 수 있습니다.

필수 구성 요소

AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항을 확인하십시오.

- 추가 기능 구독이 포함된 Amazon EKS 클러스터 계정

- AWS Marketplace에 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

AWS용 **Trident** 추가 기능을 활성화하세요

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 구성할 클러스터의 이름을 선택하십시오.
4. * 추가 기능 * 을 선택한 다음 * 추가 기능 더 보기 * 를 선택합니다.
5. 추가 기능을 선택하려면 다음 단계를 따르십시오.
   a. **AWS Marketplace** 애드온 섹션으로 스크롤하여 검색 상자에 *"Trident"*를 입력하세요.
   b. Trident by NetApp 상자의 오른쪽 상단에 있는 확인란을 선택하십시오.
   c. * 다음 * 을 선택합니다.
6. 선택한 애드온 구성 설정 페이지에서 다음을 수행하십시오.

   Pod Identity 연결을 사용하는 경우 이 단계를 건너뛰십시오.

   a. 사용할 *버전*을 선택합니다.
   b. IRSA 인증을 사용하는 경우 선택적 구성 설정에서 사용 가능한 구성 값을 설정했는지 확인하십시오.
      - 사용할 *버전*을 선택합니다.
      - **Add-on configuration schema***를 따르고 *구성 값 섹션의 **configurationValues** 매개 변수를 이전 단계에서 생성한 role-arn으로 설정합니다(값은 다음 형식이어야 합니다).

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
충돌 해결 방법으로 Override를 선택하면 기존 애드온 소프트웨어의 설정 중 하나 이상이 Amazon EKS 애드온 소프트웨어 설정으로 덮어쓰여질 수 있습니다. 이 옵션을 사용하지 않고 기존 설정과 충돌이 발생하면 작업이 실패합니다. 오류 메시지를 사용하여 충돌 문제를 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 애드온 소프트웨어가 사용자가 직접 관리해야 하는 설정을 관리하지 않는지 확인하십시오.

7. * 다음 * 을 선택합니다.
8. 검토 및 추가 페이지에서 * 생성 * 을 선택합니다.

   애드온 소프트웨어 설치가 완료되면 설치된 애드온 소프트웨어를 확인할 수 있습니다.

AWS CLI

1. `add-on.json` 파일 생성:

Pod Identity의 경우 다음 형식을 사용하십시오:

다음을 사용하세요

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

IRSA 인증의 경우 다음 형식을 사용하십시오:

```
{
  "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"
  }
}
```

`<role ARN>`을 이전 단계에서 생성한 역할의 ARN으로 바꿉니다.

2. Trident EKS 애드온을 설치합니다.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

다음 예시 명령은 Trident EKS 애드온을 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 애드온을 업데이트할 클러스터의 이름을 선택합니다.
4. 애드온 탭을 선택합니다.
5. *Trident by NetApp*을 선택한 다음 *편집*을 선택합니다.
6. **NetApp**에서 **Trident** 구성 페이지에서 다음을 수행합니다.
   a. 사용할 *버전*을 선택합니다.
   b. *선택적 구성 설정*을 확장하고 필요에 따라 수정하십시오.
   c. *변경 사항 저장*을 선택합니다.

AWS CLI

다음 예에서는 EKS 애드온을 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>'\"}"
```

eksctl

- FSxN Trident CSI 애드온의 현재 버전을 확인하십시오. `my-cluster`를 클러스터 이름으로 바꾸십시오.

  ```
  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 단계 출력의 UPDATE AVAILABLE에 반환된 버전으로 애드온 소프트웨어를 업데이트하십시오.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

`--force` 옵션을 제거하고 Amazon EKS 애드온 설정이 기존 설정과 충돌하는 경우 Amazon EKS 애드온 업데이트가 실패하며 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 지정하기 전에 Amazon EKS 애드온이 관리해야 하는 설정을 관리하지 않는지 확인하십시오. 이 옵션을 사용하면 해당 설정이 덮어쓰여지기 때문입니다. 이 설정의 다른 옵션에 대한 자세한 내용은 link:https://eksctl.io/usage/addons/["애드온"]을 참조하십시오. Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 link:https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-field-management.html["Kubernetes 필드 관리"]을 참조하십시오.

Trident EKS 애드온 제거/삭제

Amazon EKS 애드온을 제거하는 두 가지 옵션이 있습니다.

- 클러스터에 애드온 소프트웨어 유지 – 이 옵션을 선택하면 Amazon EKS에서 설정 관리가 제거됩니다. 또한 Amazon EKS에서 업데이트 알림을 보내거나 업데이트를 시작한 후 Amazon EKS 애드온을 자동으로 업데이트하는 기능도 제거됩니다. 하지만 클러스터에 애드온 소프트웨어는 유지됩니다. 이 옵션을 사용하면 애드온이 Amazon EKS 애드온이 아닌 자체 관리형 설치로 작동합니다. 이 옵션을 사용해도 애드온에 대한 다운타임은 발생하지 않습니다. 애드온을 유지하려면 명령에 `--preserve` 옵션을 포함하십시오.
- 클러스터에서 애드온 소프트웨어를 완전히 제거하기 – NetApp에서는 클러스터에 해당 애드온에 의존하는 리소스가 없는 경우에만 Amazon EKS 애드온을 클러스터에서 제거할 것을 권장합니다. 애드온을 제거하려면 `--preserve` 옵션을 `delete` 명령에서 제거하세요.

애드온 소프트웨어에 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 애드온을 제거할 클러스터의 이름을 선택하십시오.
4. **Add-ons** 탭을 선택한 다음 *Trident by NetApp*을 선택합니다.
5. *제거*를 선택합니다.
6. **netapp_trident-operator** 제거 확인 대화 상자에서 다음을 수행합니다.
   a. Amazon EKS에서 애드온 설정을 더 이상 관리하지 않도록 하려면 *클러스터에 유지*를 선택하십시오. 클러스터에 애드온 소프트웨어를 유지하여 애드온의 모든 설정을 직접 관리하려면 이 옵션을 선택하십시오.
   b. *netapp_trident-operator*를 입력합니다.
   c. *제거*를 선택합니다.

AWS CLI

`my-cluster`를 클러스터 이름으로 바꾼 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

스토리지 백엔드 구성

ONTAP SAN 및 NAS 드라이버 통합

스토리지 백엔드를 생성하려면 JSON 또는 YAML 형식의 구성 파일을 만들어야 합니다. 이 파일에는 사용할 스토리지 유형(NAS 또는 SAN), 파일 시스템, 스토리지를 가져올 SVM, 그리고 인증 방법을 지정해야 합니다. 다음 예는 NAS 기반 스토리지를 정의하고 AWS 시크릿을 사용하여 사용할 SVM에 대한 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 Backend Configuration(TBC)을 생성하고 검증합니다.

- YAML 파일에서 Trident 백엔드 구성(TBC)을 생성하고 다음 명령을 실행하십시오.

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Trident 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 Trident를 Amazon FSx for NetApp ONTAP과 통합할 수 있습니다.

- `ontap-san`: 프로비저닝된 각 PV는 자체 Amazon FSx for NetApp ONTAP 볼륨 내의 LUN입니다. 블록 스토리지에 권장됩니다.
- `ontap-nas`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨입니다. NFS 및 SMB에 권장됩니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 LUN 수를 가진 LUN입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 큐트리이며, Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 큐트리 개수를 가집니다.
- `ontap-nas-flexgroup`: 프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP FlexGroup 볼륨입니다.

드라이버 세부 정보는 "NAS 드라이버" 및 "SAN 드라이버"을 참조하십시오.

구성 파일이 생성되면 다음 명령을 실행하여 EKS 내에 생성하십시오.

```
kubectl create -f configuration_file
```

상태를 확인하려면 다음 명령을 실행합니다.

```
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 예

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	예
`version`		항상 1
`storageDriverName`	스토리지 드라이버의 이름	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
`managementLIF`	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. 정규화된 도메인 이름(FQDN)도 지정할 수 있습니다. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용할 수 있도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다. 만약 `fsxFilesystemID`를 `aws` 필드에 제공하면 `managementLIF`를 제공할 필요가 없습니다. `Trident`가 AWS에서 SVM `managementLIF` 정보를 가져오기 때문입니다. 따라서 SVM(예: vsadmin) 아래의 사용자에 대한 자격 증명을 제공해야 하며, 해당 사용자는 `vsadmin` 역할을 가지고 있어야 합니다.	"10.0.0.1", "[2001:1234:abcd::fefe]"

매개변수	설명	예
`dataLIF`	프로토콜 LIF의 IP 주소입니다. **ONTAP NAS** 드라이버: NetApp에서는 dataLIF를 지정하는 것을 권장합니다. 지정하지 않으면 Trident가 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 FQDN(정규화된 도메인 이름)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 생성할 수 있습니다. 초기 설정 후 변경할 수 있습니다. 을 참조하십시오. **ONTAP SAN** 드라이버: iSCSI의 경우 지정하지 마십시오. Trident는 ONTAP Selective LUN Map을 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. dataLIF가 명시적으로 정의된 경우 경고가 생성됩니다. Trident가 IPv6 플래그를 사용하여 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호 안에 정의해야 합니다.	
`autoExportPolicy`	자동 엑스포트 정책 생성 및 업데이트를 활성화합니다[Boolean]. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident에서 엑스포트 정책을 자동으로 관리할 수 있습니다.	`false`
`autoExportCIDRs`	`autoExportPolicy`이 활성화된 경우 `Kubernetes 노드 IP`를 필터링하는 데 사용할 `CIDR` 목록입니다. `autoExportPolicy` 및 `autoExportCIDRs` 옵션을 사용하면 Trident가 내보내기 정책을 자동으로 관리할 수 있습니다.	"["0.0.0.0/0", "::/0"]"
`labels`	볼륨에 적용할 임의의 JSON 형식 레이블 세트	""
`clientCertificate`	클라이언트 인증서의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다	""
`clientPrivateKey`	클라이언트 개인 키를 Base64로 인코딩한 값입니다. 인증서 기반 인증에 사용됩니다.	""

매개변수	설명	예
`trustedCACertificate`	신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 선택 사항입니다. 인증서 기반 인증에 사용됩니다.	""
`username`	클러스터 또는 SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. 예: vsadmin.	
`password`	클러스터 또는 SVM에 연결하는 데 사용되는 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다.	
`svm`	사용할 스토리지 가상 머신	SVM 관리 LIF가 지정된 경우 파생됩니다.
`storagePrefix`	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 생성 후에는 수정할 수 없습니다. 이 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`trident`
`limitAggregateUsage`	**Amazon FSx for NetApp ONTAP**에는 지정하지 마십시오**.** 제공된 `fsxadmin` 및 `vsadmin`에는 Trident를 사용하여 애그리게이트 사용량을 검색하고 제한하는 데 필요한 권한이 포함되어 있지 않습니다.	사용하지 마십시오.
`limitVolumeSize`	요청된 볼륨 크기가 이 값을 초과하면 프로비저닝이 실패합니다. 또한 관리하는 qtree 및 LUN의 최대 크기를 제한하며, `qtreesPerFlexvol` 옵션을 통해 FlexVol 볼륨당 최대 qtree 수를 사용자 지정할 수 있습니다.	"" (기본적으로 적용되지 않음)
`lunsPerFlexvol`	FlexVol 볼륨당 최대 LUN 수는 [50, 200] 범위 내에 있어야 합니다. SAN 전용입니다.	“100”
`debugTraceFlags`	문제 해결 시 사용할 디버그 플래그입니다. 예, {"api":false, "method":true} 문제 해결 중이거나 자세한 로그 덤프가 필요한 경우가 아니면 `debugTraceFlags` 사용하지 마십시오.	null

매개변수	설명	예
nfsMountOptions	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용합니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident는 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
nasType	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 nfs, smb 또는 null입니다. **SMB** 볼륨의 경우 **`smb`**로 설정해야 합니다**.** null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다.	nfs
qtreesPerFlexvol	FlexVol 볼륨당 최대 Qtree 수는 [50, 300] 범위 내에 있어야 합니다.	"200"
smbShare	다음 중 하나를 지정할 수 있습니다. Microsoft Management Console 또는 ONTAP CLI를 사용하여 생성한 SMB 공유의 이름 또는 Trident가 SMB 공유를 생성할 수 있도록 허용하는 이름입니다. 이 매개 변수는 Amazon FSx for NetApp ONTAP 백엔드에 필수입니다.	smb-share
useREST	ONTAP REST API를 사용하기 위한 부울 매개 변수입니다. true`로 설정하면 Trident는 ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할은 `ontap 애플리케이션에 대한 액세스 권한이 있어야 합니다. 이는 사전 정의된 vsadmin 및 cluster-admin 역할로 충족됩니다.	false
aws	AWS FSx for ONTAP의 구성 파일에서 다음을 지정할 수 있습니다. - fsxFilesystemID: AWS FSx 파일 시스템의 ID를 지정합니다. - apiRegion: AWS API 리전 이름입니다. - apikey: AWS API 키입니다. - secretKey: AWS 비밀 키입니다.	"" "" ""

매개변수	설명	예
`credentials`	AWS Secrets Manager에 저장할 FSx SVM 자격 증명을 지정합니다. - `name`: SVM 자격 증명이 포함된 시크릿의 Amazon 리소스 이름(ARN)입니다. - `type`: `awsarn`로 설정합니다. 자세한 내용은 "AWS Secrets Manager 암호를 생성합니다"을 참조하십시오.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

구성의 `defaults` 섹션에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다. 예를 들어, 아래 구성 예를 참조하십시오.

매개변수	설명	기본값
`spaceAllocation`	LUN의 공간 할당	`true`
`spaceReserve`	공간 예약 모드, "none"(씬) 또는 "volume"(씩)	`none`
`snapshotPolicy`	사용할 스냅샷 정책	`none`
`qosPolicy`	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드당 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택합니다. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9.8 이상이 필요합니다. 공유되지 않는 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 상한선을 적용합니다.	""
`adaptiveQosPolicy`	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드별로 qosPolicy 또는 adaptiveQosPolicy 중 하나를 선택하십시오. ontap-nas-economy에서는 지원되지 않습니다.	""
`snapshotReserve`	스냅샷용으로 예약된 볼륨의 비율 "0"	`snapshotPolicy`이 `none`인 경우, `else` ""
`splitOnClone`	생성 시 상위 항목에서 클론 분할	`false`

매개변수	설명	기본값
`encryption`	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"	`false`
`luksEncryption`	LUKS 암호화를 활성화합니다. "Linux Unified Key Setup(LUKS) 사용"를 참조하십시오. SAN에만 해당됩니다.	""
`tieringPolicy`	사용할 계층화 정책 `none`	
`unixPermissions`	새 볼륨 모드입니다. **SMB** 볼륨의 경우 비워 두십시오.	""
`securityStyle`	새 볼륨에 대한 보안 스타일. NFS는 `mixed` 및 `unix` 보안 스타일을 지원합니다. SMB는 `mixed` 및 `ntfs` 보안 스타일을 지원합니다.	NFS 기본값은 `unix`입니다. SMB 기본값은 `ntfs`입니다.

SMB 볼륨 프로비저닝

`ontap-nas` 드라이버를 사용하여 `SMB` 볼륨을 프로비저닝할 수 있습니다. `<<ONTAP SAN 및 NAS 드라이버 통합>>`를 완료하기 전에 다음 단계를 완료하십시오: `link:https://docs.netapp.com/us-en/trident/trident-use/worker-node-prep.html#prepare-to-provision-smb-volumes["SMB 볼륨 프로비저닝 준비"].`

스토리지 클래스 및 PVC 구성

Kubernetes StorageClass 오브젝트를 구성하고 스토리지 클래스를 생성하여 Trident에 볼륨을 프로비저닝하는 방법을 지시합니다. 구성된 Kubernetes StorageClass를 사용하는 PersistentVolumeClaim(PVC)을 생성하여 PV에 대한 액세스를 요청합니다. 그런 다음 PV를 파드에 마운트할 수 있습니다.

스토리지 클래스를 생성합니다

Kubernetes StorageClass 오브젝트 구성

https://kubernetes.io/docs/concepts/storage/storage-classes/["Kubernetes StorageClass 오브젝트"^] 객체는 해당 클래스에 사용되는 프로비저너로 Trident를 식별하고 Trident에 볼륨을 프로비저닝하는 방법을 지시합니다. 이 예제를 사용하여 NFS를 사용하는 볼륨에 대한 Storageclass를 설정하십시오(전체 속성 목록은 아래 Trident 속성 섹션을 참조하십시오):

```
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 볼륨용 Storageclass를 설정하려면 다음 예를 사용하십시오.

```
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 스토리지 클래스에 필요한 `mountOptions`을(를) 추가하십시오.

```
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
```

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

스토리지 클래스를 생성합니다

단계

1. 이것은 Kubernetes 객체이므로 `kubectl`을 사용하여 Kubernetes에서 생성합니다.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스가 표시되어야 하며, Trident가 백엔드에서 풀을 검색했을 것입니다.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

PVC를 생성합니다

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

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

PVC를 생성한 후에는 볼륨을 포드에 마운트할 수 있습니다.

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

이 예에서는 기본 PVC 구성 옵션을 보여 줍니다.

RWX 액세스가 있는 **PVC**

이 예제는 RWX 액세스 권한이 있는 기본 PVC를 보여주며, 이는 StorageClass라는 이름 `basic-csi`과 연결되어 있습니다.

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

iSCSI를 사용하는 **PVC** 예

이 예제는 RWO 액세스가 가능한 iSCSI용 기본 PVC를 보여주며, 이는 StorageClass라는 이름의 `protection-gold`와(과) 연결되어 있습니다.

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

PVC 생성

단계

1. PVC를 생성합니다.

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

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi       RWO                          5m
```

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

Trident 특성

이러한 매개변수는 특정 유형의 볼륨을 프로비저닝하는 데 사용할 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공	요청	지원 대상:
미디어[1]	문자열	HDD, 하이브리드, SSD	풀에는 이러한 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다를 의미합니다	지정된 미디어 유형	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	문자열	씬, 씩	풀은 이 프로비저닝 방식을 지원합니다	프로비저닝 방법이 지정되었습니다	thick: 모든 ONTAP, thin: 모든 ONTAP 및 SolidFire-SAN
backendType	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀은 이러한 유형의 백엔드에 속합니다	지정된 백엔드	모든 드라이버
스냅샷	불	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
클론	불	참, 거짓	풀은 볼륨 복제를 지원합니다	클론이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san

속성	유형	값	제공	요청	지원 대상:
암호화	불	참, 거짓	스토리지 풀은 암호화된 볼륨을 지원합니다.	암호화가 설정된 볼륨	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	양의 정수	풀은 이 범위의 IOPS를 보장할 수 있습니다	볼륨에서 이러한 IOPS를 보장합니다	SolidFire-SAN

[1]: ONTAP Select 시스템에서 지원되지 않습니다

샘플 애플리케이션 배포

스토리지 클래스와 PVC가 생성되면 PV를 파드에 마운트할 수 있습니다. 이 섹션에서는 PV를 파드에 연결하는 예제 명령과 구성을 설명합니다.

단계

1. POD에 볼륨을 마운트합니다.

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

다음 예시는 PVC를 포드에 연결하는 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
```

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

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

```
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 클러스터에 **Trident EKS** 애드온을 구성합니다

NetApp Trident는 Kubernetes 환경에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 배포에 집중할 수 있도록 지원합니다. NetApp Trident EKS 애드온에는 최신 보안 패치와 버그 수정 사항이 포함되어 있으며, AWS에서 Amazon EKS와의 호환성을 검증받았습니다. 이 EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 유지할 수 있으며, 애드온 설치, 구성 및 업데이트에 필요한 작업량을 줄일 수 있습니다.

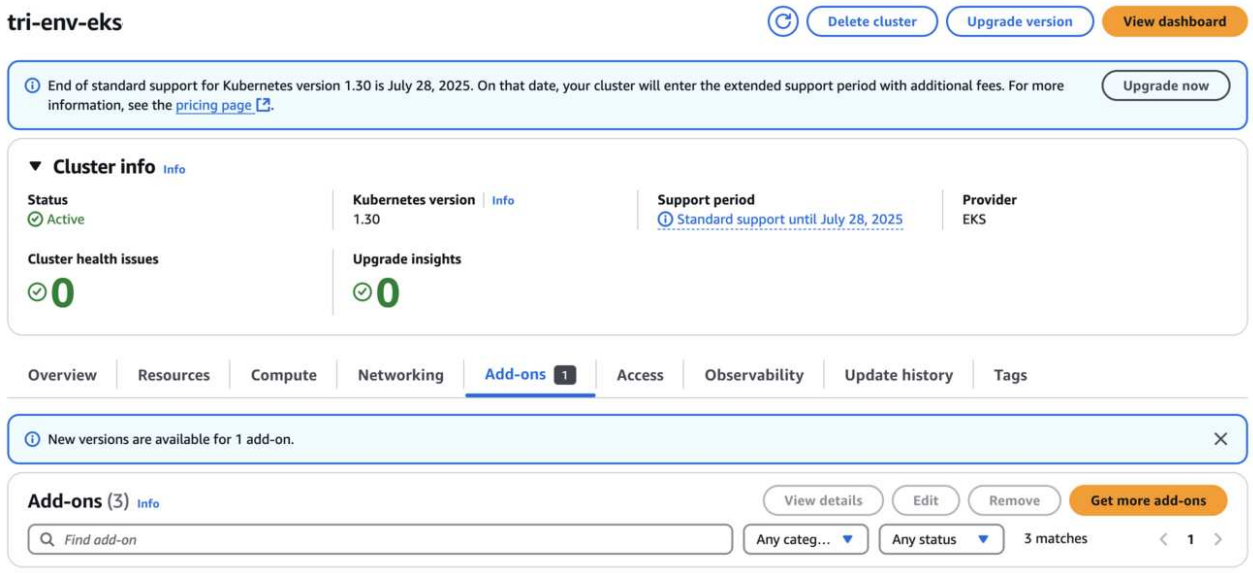
필수 구성 요소

AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항을 확인하십시오.

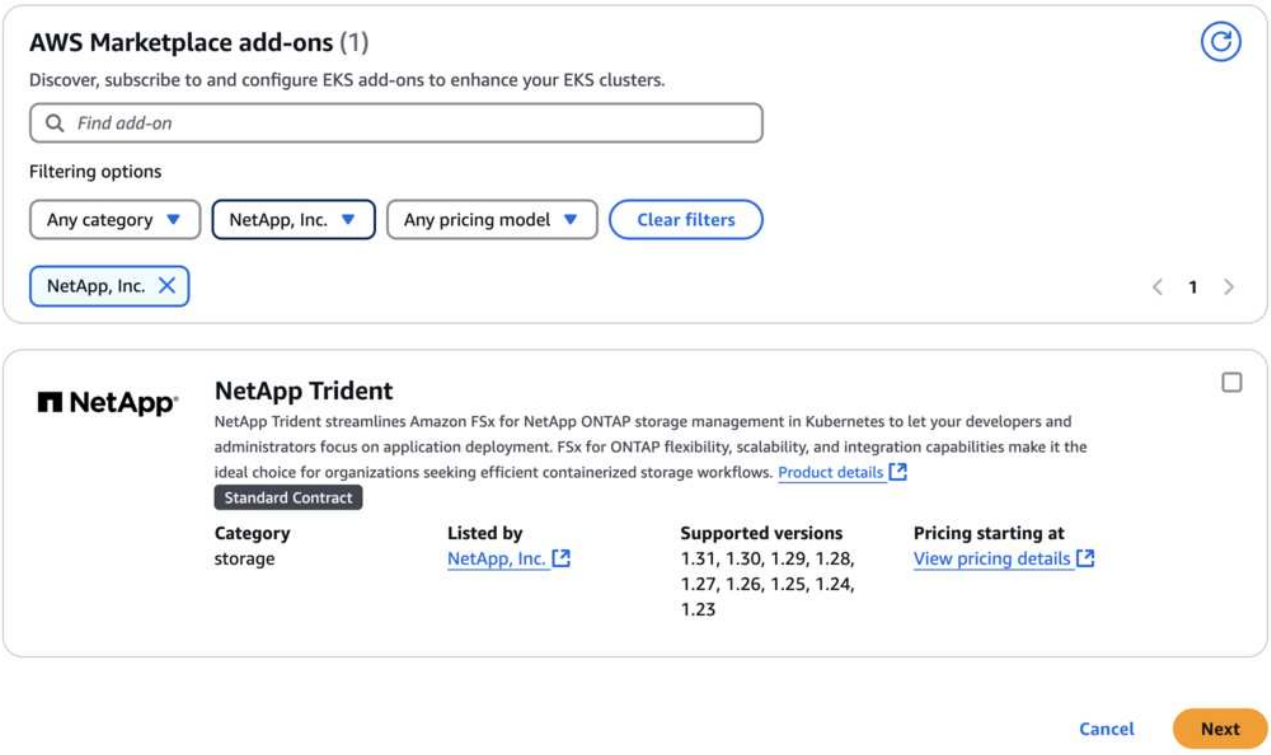
- 추가 기능을 사용할 수 있는 권한이 있는 Amazon EKS 클러스터 계정입니다. "Amazon EKS 추가 기능"을(를) 참조하십시오.
- AWS Marketplace에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2(AL2_x86_64) 또는 Amazon Linux 2 Arm(AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

단계

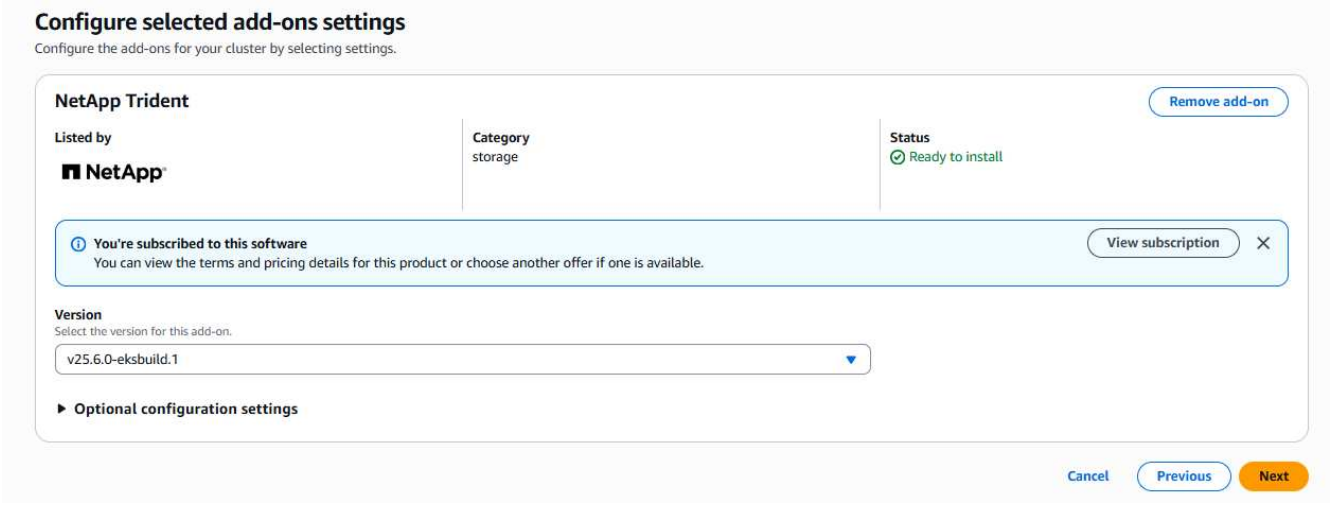
1. EKS Pod가 AWS 리소스에 액세스할 수 있도록 IAM 역할과 AWS 시크릿을 생성해야 합니다. 자세한 지침은 "IAM 역할 및 AWS Secret을 생성합니다"를 참조하십시오.
2. EKS Kubernetes 클러스터에서 **Add-ons** 탭으로 이동합니다.



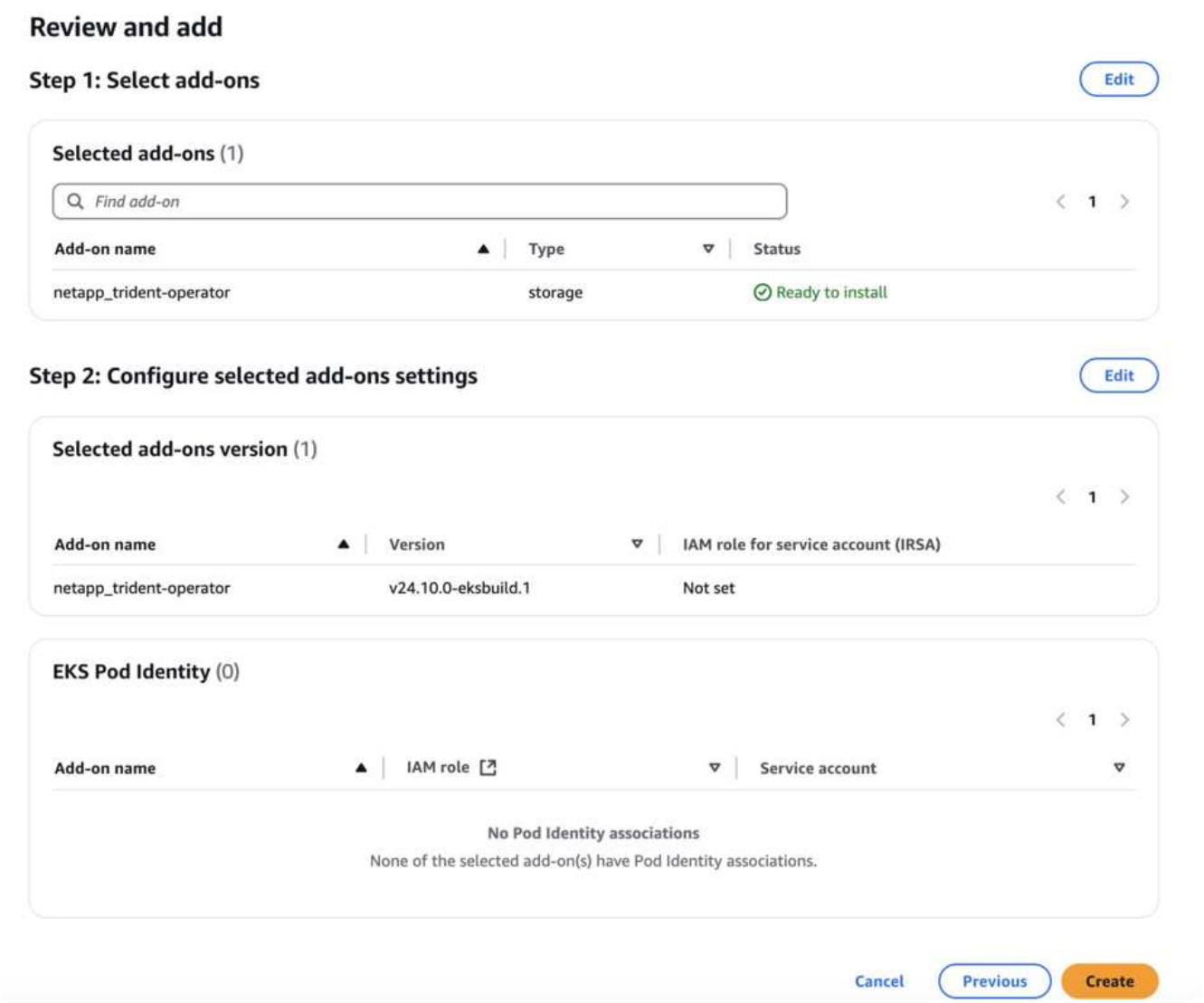
3. *AWS Marketplace add-ons*로 이동하여 *storage* 카테고리를 선택하세요.



4. *NetApp Trident*를 찾아 Trident 추가 기능 확인란을 선택한 다음 *다음*을 클릭합니다.
5. 원하는 추가 기능 버전을 선택합니다.

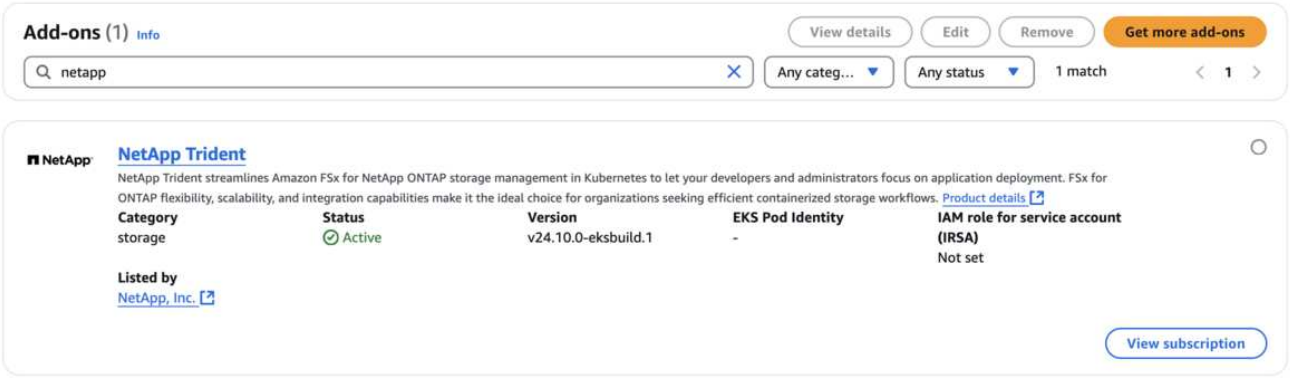


6. 필요한 추가 기능 설정을 구성하십시오.



7. IRSA(서비스 계정용 IAM 역할)를 사용하는 경우 추가 구성 단계"여기"를 참조하십시오.
8. *생성*을 선택합니다.

9. 추가 기능의 상태가 _Active_인지 확인하십시오.



10. 다음 명령을 실행하여 Trident가 클러스터에 제대로 설치되었는지 확인하십시오.

```
kubectl get pods -n trident
```

11. 설정을 계속 진행하여 스토리지 백엔드를 구성하십시오. 자세한 내용은 "스토리지 백엔드 구성"를 참조하십시오.

CLI를 사용하여 **Trident EKS** 애드온 설치/제거

CLI를 사용하여 **NetApp Trident EKS** 추가 기능을 설치합니다.

다음 예시 명령어는 Trident EKS 애드온을 설치합니다.

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.0-eksbuild.1(전용 버전 포함)
```

다음 예시 명령은 Trident EKS 애드온 버전 25.6.1를 설치합니다:

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.1-eksbuild.1 (전용 버전 포함)
```

다음 예시 명령은 Trident EKS 애드온 버전 25.6.2를 설치합니다:

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.2-eksbuild.1 (전용 버전 포함)
```

CLI를 사용하여 **NetApp Trident EKS** 추가 기능을 제거합니다.

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

kubectl을 사용하여 백엔드 생성

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. 백엔드는 Trident가 해당 스토리지 시스템과 통신하는 방법과 Trident가 해당 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Trident 설치 후 다음 단계는 백엔드를 생성하는 것입니다.
`TridentBackendConfig` Custom Resource Definition(CRD)을 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 생성하고 관리할 수 있습니다. `kubectl` 또는

Kubernetes 배포판에 맞는 CLI 도구를 사용하여 수행할 수 있습니다.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`)는 프런트엔드 네임스페이스 CRD로, `kubectl`를 사용하여 `Trident` 백엔드를 관리할 수 있도록 해줍니다. 이제 `Kubernetes` 및 스토리지 관리자는 별도의 명령줄 유틸리티(`tridentctl` 없이 Kubernetes CLI를 통해 백엔드를 직접 생성하고 관리할 수 있습니다.

`TridentBackendConfig` 객체가 생성될 때 다음과 같은 일이 발생합니다.

- 백엔드는 사용자가 제공하는 구성을 기반으로 Trident에 의해 자동으로 생성됩니다. 이는 내부적으로 `TridentBackend` (`tbe`, `tridentbackend`) CR로 표현됩니다.
- `TridentBackendConfig`는 Trident에 의해 생성된 `TridentBackend`에 고유하게 바인딩됩니다.

각 `TridentBackendConfig`은 `TridentBackend`와 일대일 매핑을 유지합니다. 전자는 백엔드를 설계하고 구성하기 위해 사용자에게 제공되는 인터페이스이며, 후자는 Trident가 실제 백엔드 객체를 표현하는 방식입니다.

`TridentBackend` CR은 Trident에 의해 자동으로 생성됩니다. CR을 수정해서는 안 됩니다. 백엔드를 업데이트하려면 `TridentBackendConfig` 객체를 수정하십시오.

`TridentBackendConfig` CR의 형식은 다음 예를 참조하십시오.

```
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
```

원하는 스토리지 플랫폼/서비스에 대한 샘플 구성은 "trident-installer" 디렉터리의 예제를 참조하십시오.

`spec`는 백엔드별 구성 매개변수를 사용합니다. 이 예에서 백엔드는 `ontap-san` 스토리지 드라이버를 사용하며 여기에 표로 정리된 구성 매개변수를 사용합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 xref:{relative_path}backends.html["스토리지 드라이버에 대한 백엔드 구성 정보"^]를 참조하십시오.

`spec` 섹션에는 또한 `credentials` 및 `deletionPolicy` 필드가 포함되어 있으며, 이는 `TridentBackendConfig` CR에서 새롭게 도입되었습니다.

- `credentials`: 이 매개변수는 필수 입력 항목이며 스토리지 시스템/서비스 인증에 사용되는 자격 증명을 포함합니다. 이 값은 사용자가 생성한 Kubernetes Secret으로 설정됩니다. 자격 증명을 일반 텍스트로 전달할 수 없으며, 그렇게 할 경우 오류가 발생합니다.
- `deletionPolicy`: 이 필드는 `TridentBackendConfig`가 삭제될 때 발생해야 하는 작업을 정의합니다. 다음 두 가지 값 중 하나를 사용할 수 있습니다.
  - `delete`: 이렇게 하면 `TridentBackendConfig` CR과 관련 백엔드가 모두 삭제됩니다. 이것이 기본값입니다.
  - `retain`: `TridentBackendConfig` CR이 삭제되더라도 백엔드 정의는 계속 남아 있으며 `tridentctl`를 사용하여 관리할 수 있습니다. 삭제 정책을 `retain`로 설정하면 사용자는 이전 릴리스(`21.04` 이전 버전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있습니다. 이 필드의 값은 `TridentBackendConfig` 생성 후 업데이트할 수 있습니다.

백엔드의 이름은 `spec.backendName`를 사용하여 설정합니다. 지정하지 않으면 백엔드 이름은 `TridentBackendConfig` 객체의 이름(metadata.name)으로 설정됩니다. `spec.backendName`를 사용하여 백엔드 이름을 명시적으로 설정하는 것이 좋습니다.

`tridentctl`로 생성된 백엔드는 연결된 `TridentBackendConfig` 오브젝트가 없습니다. 이러한 백엔드는 `kubectl`를 사용하여 `TridentBackendConfig` CR을 생성함으로써 관리할 수 있습니다. `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` 등과 같은 동일한 구성 파라미터를 지정해야 하므로 주의가 필요합니다. Trident는 새로 생성된 `TridentBackendConfig`를 기존 백엔드와 자동으로 바인딩합니다.

단계 개요

`kubectl`를 사용하여 새 백엔드를 생성하려면 다음을 수행해야 합니다.

1. "Kubernetes Secret"을 생성합니다. 이 secret에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. `TridentBackendConfig` 객체를 생성합니다. 이 객체에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되며 이전 단계에서 생성한 비밀 키를 참조합니다.

백엔드를 생성한 후 `kubectl get tbc <tbc-name> -n <trident-namespace>`를 사용하여 상태를 확인하고 추가 세부 정보를 수집할 수 있습니다.

1단계**: Kubernetes Secret** 생성

백엔드에 대한 액세스 자격 증명이 포함된 Secret을 생성합니다. 이는 각 스토리지 서비스/플랫폼마다 고유합니다. 다음은 예입니다.

```
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 스토리지 플랫폼의 Secret에 포함되어야 하는 필드를 요약한 것입니다.

스토리지 플랫폼 **Secret Fields** 설명	비밀	필드 설명
Azure NetApp Files	clientID	앱 등록의 클라이언트 ID
요소(NetApp HCI/SolidFire)	엔드포인트	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP
ONTAP	사용자 이름	클러스터/SVM에 연결하는 데 사용할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다.
ONTAP	비밀번호	클러스터/SVM에 연결하는 데 사용되는 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다.
ONTAP	clientPrivateKey	클라이언트 개인 키의 Base64 인코딩 값입니다. 인증서 기반 인증에 사용됩니다.
ONTAP	chapUsername	인바운드 사용자 이름. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`

스토리지 플랫폼 **Secret Fields** 설명	비밀	필드 설명
ONTAP	chapInitiatorSecret	CHAP 개시자 비밀 키. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`
ONTAP	chapTargetUsername	대상 사용자 이름. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 대상 시작자 비밀 키. useCHAP=true인 경우 필수입니다. `ontap-san` 및 `ontap-san-economy`

이 단계에서 생성된 Secret은 다음 단계에서 생성되는 `spec.credentials` 객체의 `TridentBackendConfig` 필드에서 참조됩니다.

2단계: `TridentBackendConfig` **CR** 생성

이제 `TridentBackendConfig` CR을 생성할 준비가 되었습니다. 이 예에서는 `ontap-san` 드라이버를 사용하는 백엔드가 아래 표시된 `TridentBackendConfig` 객체를 사용하여 생성됩니다.

```
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: `TridentBackendConfig` **CR**의 상태를 확인합니다

`TridentBackendConfig` CR을 생성했으므로 이제 상태를 확인할 수 있습니다. 다음 예시를 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 `TridentBackendConfig` CR에 바인딩되었습니다.

Phase는 다음 값 중 하나를 사용할 수 있습니다.

- `Bound`: `TridentBackendConfig` CR은 백엔드와 연결되어 있으며, 해당 백엔드에는 `configRef`가 `TridentBackendConfig` CR의 uid로 설정되어 있습니다.
- `Unbound`: `""`로 표현됩니다. `TridentBackendConfig` 객체는 백엔드에 바인딩되지 않았습니다. 새로 생성되는 모든 `TridentBackendConfig` CR은 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 Unbound 상태로 되돌릴 수 없습니다.
- `Deleting`: `TridentBackendConfig`CR의 `deletionPolicy` 삭제가 설정되었습니다. `TridentBackendConfig`CR이 삭제되면 삭제 중 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVC)이 없는 경우 `TridentBackendConfig`를 삭제하면 `Trident`가 백엔드와 `TridentBackendConfig` CR을 삭제합니다.
  - 백엔드에 하나 이상의 PVC가 존재하면 삭제 상태로 전환됩니다. `TridentBackendConfig` CR도 이후 삭제 단계로 들어갑니다. 백엔드와 `TridentBackendConfig`는 모든 PVC가 삭제된 후에만 삭제됩니다.
- `Lost`: `TridentBackendConfig` CR과 연결된 백엔드가 실수로 또는 의도적으로 삭제되었지만 `TridentBackendConfig` CR에는 여전히 삭제된 백엔드에 대한 참조가 남아 있습니다. `TridentBackendConfig` CR은 `deletionPolicy` 값과 관계없이 삭제할 수 있습니다.
- `Unknown`: Trident는 `TridentBackendConfig` CR과 연결된 백엔드의 상태 또는 존재 여부를 확인할 수 없습니다. 예를 들어 API 서버가 응답하지 않거나 `tridentbackends.trident.netapp.io` CRD가 없는 경우입니다. 이 경우 개입이 필요할 수 있습니다.

이 단계에서 백엔드가 성공적으로 구축되었습니다! "백엔드 업데이트 및 백엔드 삭제"와 같이 추가로 처리할 수 있는 몇 가지 작업이 있습니다.

(선택 사항) 4단계: 자세한 정보 확인

다음 명령을 실행하여 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 `TridentBackendConfig`의 YAML/JSON 덤프도 얻을 수 있습니다.

```
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`에는 `backendName` 및 `backendUUID`가 해당 `TridentBackendConfig` CR에 대한 응답으로 생성된 백엔드의 정보가 포함되어 있습니다. `lastOperationStatus` 필드는 `TridentBackendConfig` CR의 마지막 작업 상태를 나타내며, 이는 사용자가 트리거한 경우(예: 사용자가 `spec`에서 무언가를 변경한 경우) 또는 `Trident`에 의해 트리거된 경우(예: `Trident` 재시작 중)에 발생할 수 있습니다. 이 값은 `Success` 또는 `Failed` 중 하나일 수 있습니다. `phase`는 `TridentBackendConfig` CR과 백엔드 간의 관계

상태를 나타냅니다. 위 예시에서, `phase`의 값은 `Bound`이며, 이는 `TridentBackendConfig` CR이 백엔드와 연결되어 있음을 의미합니다.

`kubectl -n trident describe tbc <tbc-cr-name>` 명령을 실행하여 이벤트 로그의 세부 정보를 확인할 수 있습니다.

연결된 `TridentBackendConfig` 객체가 포함된 백엔드는 `tridentctl`를 사용하여 업데이트하거나 삭제할 수 없습니다. `tridentctl`와 `TridentBackendConfig` 간 전환에 필요한 단계들을 이해하려면 "여기를 참조하십시오".

백엔드 관리

kubectl을 사용하여 백엔드 관리를 수행합니다

`kubectl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 삭제합니다

`TridentBackendConfig`를 삭제하면 `Trident`에 백엔드를 삭제/유지하도록(`deletionPolicy`에 따라) 지시하는 것입니다. 백엔드를 삭제하려면 `deletionPolicy`가 `delete`로 설정되어 있는지 확인하십시오. `TridentBackendConfig`만 삭제하려면 `deletionPolicy`가 `retain`으로 설정되어 있는지 확인하십시오. 이렇게 하면 백엔드가 계속 존재하며 `tridentctl`를 사용하여 관리할 수 있습니다.

다음 명령을 실행합니다.

```
kubectl delete tbc <tbc-name> -n trident
```

Trident는 `TridentBackendConfig`에서 사용 중인 Kubernetes Secret을 삭제하지 않습니다. Kubernetes 사용자가 Secret 정리를 담당합니다. Secret을 삭제할 때는 주의해야 합니다. 백엔드에서 사용하지 않는 Secret만 삭제해야 합니다.

기존 백엔드 보기

다음 명령을 실행합니다.

```
kubectl get tbc -n trident
```

`tridentctl get backend -n trident` 또는 `tridentctl get backend -o yaml -n trident`를 실행하여 존재하는 모든 백엔드 목록을 얻을 수도 있습니다. 이 목록에는 `tridentctl`로 생성된 백엔드도 포함됩니다.

백엔드 업데이트

백엔드를 업데이트해야 하는 이유는 여러 가지가 있을 수 있습니다.

- 스토리지 시스템에 대한 자격 증명이 변경되었습니다. 자격 증명을 업데이트하려면 `TridentBackendConfig` 객체에 사용되는 Kubernetes Secret을 업데이트해야 합니다. Trident는 제공된 최신 자격 증명으로 백엔드를 자동으로 업데이트합니다. 다음 명령을 실행하여 Kubernetes Secret을 업데이트하십시오.

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- 매개변수(예: 사용 중인 ONTAP SVM의 이름)를 업데이트해야 합니다.
  - 다음 명령을 사용하여 Kubernetes를 통해 `TridentBackendConfig` 객체를 직접 업데이트할 수 있습니다.

```
kubectl apply -f <updated-backend-file.yaml>
```

  - 또는 다음 명령을 사용하여 기존 `TridentBackendConfig` CR을 변경할 수 있습니다.

```
kubectl edit tbc <tbc-name> -n trident
```

- 백엔드 업데이트가 실패하면 백엔드는 마지막으로 알려진 구성을 계속 유지합니다. 로그를 확인하여 원인을 파악하려면 `kubectl get tbc <tbc-name> -o yaml -n trident` 또는 `kubectl describe tbc <tbc-name> -n trident`을 실행하십시오.
- 구성 파일의 문제를 식별하고 수정한 후 업데이트 명령을 다시 실행할 수 있습니다.

tridentctl을 사용하여 백엔드 관리를 수행합니다

`tridentctl`를 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드 생성

"백엔드 구성 파일"를 생성한 후 다음 명령을 실행하십시오.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 `create` 명령을 다시 실행하기만 하면 됩니다.

백엔드를 삭제합니다

Trident에서 백엔드를 삭제하려면 다음 단계를 따르세요.

1. 백엔드 이름 검색:

```
tridentctl get backend -n trident
```

2. 백엔드 삭제:

```
tridentctl delete backend <backend-name> -n trident
```

> Trident가 이 백엔드에서 프로비저닝한 볼륨과 스냅샷이 아직 남아 있는 경우, 백엔드를 삭제하면 해당 백엔드에서 새 볼륨을 프로비저닝할 수 없습니다. 백엔드는 "Deleting" 상태로 계속 유지됩니다.

기존 백엔드 보기

Trident가 알고 있는 백엔드를 보려면 다음을 수행합니다.

- 요약을 보려면 다음 명령을 실행하십시오.

```
tridentctl get backend -n trident
```

- 모든 세부 정보를 확인하려면 다음 명령을 실행하십시오.

```
tridentctl get backend -o json -n trident
```

백엔드 업데이트

새 백엔드 구성 파일을 생성한 후 다음 명령을 실행하십시오.

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

백엔드 업데이트가 실패하면 백엔드 구성에 문제가 있거나 유효하지 않은 업데이트를 시도한 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 식별하고 수정한 후 update 명령을 다시 실행하기만 하면 됩니다.

백엔드를 사용하는 스토리지 클래스를 식별합니다

이는 백엔드 객체에 대해 tridentctl`이(가) 출력하는 JSON을 사용하여 답변할 수 있는 질문 유형의 예입니다. 이는 설치해야 하는 `jq 유틸리티를 사용합니다.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

이는 `TridentBackendConfig`을 사용하여 생성된 백엔드에도 적용됩니다.

백엔드 관리 옵션 간 이동

Trident에서 백엔드를 관리하는 다양한 방법에 대해 알아보십시오.

백엔드 관리 옵션

`TridentBackendConfig`의 도입으로 관리자는 이제 백엔드를 관리하는 두 가지 고유한 방법을 갖게 되었습니다. 이로 인해 다음과 같은 질문이 제기됩니다.

- `tridentctl`를 사용하여 생성된 백엔드를 `TridentBackendConfig`로 관리할 수 있습니까?
- `TridentBackendConfig`를 사용하여 생성된 백엔드를 `tridentctl`를 사용하여 관리할 수 있습니까?

tridentctl`를 사용하여 백엔드 관리 `TridentBackendConfig

이 섹션에서는 Kubernetes 인터페이스를 통해 직접 tridentctl`를 사용하여 생성된 백엔드를 관리하기 위해 `TridentBackendConfig 오브젝트를 생성하는 데 필요한 단계를 다룹니다.

다음 시나리오에 적용됩니다.

- 기존 백엔드의 경우, `TridentBackendConfig`가 없으며, 이는 `tridentctl`로 생성되었기 때문입니다.
- tridentctl`로 생성된 새 백엔드, 다른 `TridentBackendConfig 객체가 존재합니다.

두 시나리오 모두에서 백엔드는 계속 존재하며 Trident는 볼륨을 예약하고 운영합니다. 관리자는 여기에서 두 가지 중 하나를 선택할 수 있습니다.

- `tridentctl`을 사용하여 생성한 백엔드를 계속 관리합니다.
- 생성된 백엔드를 tridentctl 새 TridentBackendConfig 오브젝트에 바인딩합니다. 이렇게 하면 백엔드는 `kubectl`를 사용하여 관리되고 `tridentctl`로는 관리되지 않습니다.

`kubectl`을 사용하여 기존 백엔드를 관리하려면 기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들어야 합니다. 다음은 그 작동 방식에 대한 개요입니다.

1. Kubernetes Secret을 생성합니다. 이 시크릿에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. `TridentBackendConfig` 개체를 만듭니다. 여기에는 스토리지 클러스터/서비스에 대한 세부 정보가 포함되어 있으며 이전 단계에서 만든 시크릿을 참조합니다. 동일한 구성 매개변수(예: `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` 등)를 지정하도록 주의해야 합니다. `spec.backendName`는 기존 백엔드의 이름으로 설정해야 합니다.

0단계**:** 백엔드 식별

기존 백엔드에 바인딩하는 `TridentBackendConfig`를 만들려면 백엔드 구성을 가져와야 합니다. 이 예에서는 다음 JSON 정의를 사용하여 백엔드를 만들었다고 가정합니다.

```
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** 생성

이 예에 표시된 대로 백엔드에 대한 자격 증명이 포함된 Secret을 생성합니다.

```
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: `TridentBackendConfig` **CR** 생성

다음 단계는 기존에 존재하는 `ontap-nas-backend`에 자동으로 바인딩되는 `TridentBackendConfig` CR을 생성하는 것입니다(이 예시와 같이). 다음 요구 사항이 충족되는지 확인하십시오:

- 동일한 백엔드 이름이 `spec.backendName`에 정의되어 있습니다.
- 구성 매개변수는 원래 백엔드와 동일합니다.
- 가상 풀(있는 경우)은 원래 백엔드에서와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우 `TridentBackendConfig`는 다음과 같이 표시됩니다:

```
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: `TridentBackendConfig` **CR의 상태를 확인합니다**

`TridentBackendConfig`을 생성한 후 해당 단계는 `Bound`이어야 합니다. 또한 기존 백엔드의 이름 및 UUID와 동일한 백엔드 이름을 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 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

이제 백엔드는 tbc-ontap-nas-backend TridentBackendConfig 객체를 사용하여 완전히 관리됩니다.

`TridentBackendConfig`를 사용하여 `tridentctl` 백엔드 관리

`tridentctl`는 `TridentBackendConfig`를 사용하여 생성된 백엔드를 나열하는 데 사용할 수 있습니다. 또한, 관리자는 `tridentctl`를 통해 이러한 백엔드를 완전히 관리하도록 선택할 수 있으며, `TridentBackendConfig`를 삭제하고 `spec.deletionPolicy`가 `retain`로 설정되어 있는지 확인할 수 있습니다.

0단계**:** 백엔드 식별

예를 들어 다음 백엔드가 `TridentBackendConfig`을 사용하여 생성되었다고 가정해 보겠습니다:

```
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 `TridentBackendConfig`이(가) 성공적으로 생성되었으며 백엔드에 바인딩된 것을 볼 수 있습니다[백엔드의 UUID 관찰].

1단계: **`deletionPolicy`**가 **`retain`**로 설정되어 있는지 확인합니다

`deletionPolicy`의 값을 살펴보겠습니다. 이것은 `retain`로 설정해야 합니다. 이렇게 하면 `TridentBackendConfig` CR이 삭제되더라도 백엔드 정의는 계속 존재하며 `tridentctl`로 관리할 수 있습니다.

```
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
```

`deletionPolicy`이(가) `retain`(으)로 설정되지 않은 경우 다음 단계로 진행하지 마십시오.

2단계: `TridentBackendConfig` **CR** 삭제

마지막 단계는 `TridentBackendConfig` CR을 삭제하는 것입니다. `deletionPolicy`가 `retain`로 설정되어 있는지 확인한 후 삭제를 진행하면 됩니다:

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

`TridentBackendConfig` 개체를 삭제하면 `Trident`는 백엔드 자체를 실제로 삭제하지 않고 단순히 제거합니다.

스토리지 클래스 생성 및 관리

스토리지 클래스를 생성합니다

Kubernetes StorageClass 오브젝트를 구성하고 Trident 볼륨 프로비저닝 방법을 알려주는 스토리지 클래스를 생성합니다.

Kubernetes StorageClass 오브젝트 구성

https://kubernetes.io/docs/concepts/storage/storage-classes/["Kubernetes StorageClass 오브젝트"^]은 Trident를 해당 클래스에 사용되는 프로비저너로 식별하고 Trident에 볼륨 프로비저닝 방법을 지시합니다. 예:

```
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
```

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

스토리지 클래스를 생성합니다

StorageClass 객체를 생성한 후 스토리지 클래스를 생성할 수 있습니다. 스토리지 클래스 샘플에서는 사용하거나 수정할 수 있는 몇 가지 기본 샘플을 제공합니다.

단계

1. 이것은 Kubernetes 객체이므로 `kubectl`을 사용하여 Kubernetes에서 생성합니다.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 **basic-csi** 스토리지 클래스가 표시되어야 하며, Trident가 백엔드에서 풀을 검색했을 것입니다.

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

스토리지 클래스 샘플

Trident는 "특정 백엔드를 위한 간단한 스토리지 클래스 정의"를 제공합니다.

또는 설치 프로그램과 함께 제공되는 `sample-input/storage-class-csi.yaml.templ` 파일을 편집하여 `*BACKEND_TYPE*`를 스토리지 드라이버 이름으로 교체할 수도 있습니다.

```
./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 클래스를 확인합니다

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행하십시오.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하십시오.

```
kubectl get storageclass <storage-class> -o json
```

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행하십시오.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행하십시오.

```
tridentctl get storageclass <storage-class> -o json
```

기본 스토리지 클래스를 설정합니다

Kubernetes 1.6에서는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 사용자가 영구 볼륨 클레임(PVC)에서 스토리지 클래스를 지정하지 않으면 이 기본 스토리지 클래스가 영구 볼륨을 프로비저닝하는 데 사용됩니다.

- 스토리지 클래스 정의에서 주석 `storageclass.kubernetes.io/is-default-class`을 true로 설정하여 기본 스토리지 클래스를 정의합니다. 사양에 따르면 다른 값이나 주석이 없으면 false로 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 스토리지 클래스 주석을 제거할 수 있습니다.

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 설치 프로그램 번들에도 이 주석이 포함된 예제가 있습니다.

> 클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes는 기술적으로 두 개 이상의 스토리지 클래스를 사용하는 것을 막지는 않지만 기본 스토리지 클래스가 전혀 없는 것처럼 작동합니다.

스토리지 클래스의 백엔드를 식별합니다

이는 `tridentctl` Trident 백엔드 객체에 대해 출력되는 JSON을 사용하여 답변할 수 있는 질문 유형의 예입니다. 이 예시에서는 `jq` 유틸리티를 사용하는데, 필요에 따라 먼저 설치해야 할 수도 있습니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

스토리지 클래스를 삭제합니다

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행하십시오.

```
kubectl delete storageclass <storage-class>
```

`<storage-class>`는 스토리지 클래스로 교체해야 합니다.

이 스토리지 클래스를 통해 생성된 영구 볼륨은 변경되지 않고 그대로 유지되며, Trident가 계속해서 해당 볼륨을 관리합니다.

Trident는 생성하는 볼륨에 대해 빈 `fsType`을 적용합니다. iSCSI 백엔드의 경우 StorageClass에서 `parameters.fsType`을 적용하는 것이 좋습니다. 기존 StorageClasses를 삭제하고 `parameters.fsType`이 지정된 상태로 다시 생성해야 합니다.

볼륨 프로비저닝 및 관리

볼륨 프로비저닝

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

개요

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

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

PVC를 생성한 후에는 볼륨을 포드에 마운트할 수 있습니다.

PVC를 생성합니다

단계

1. PVC를 생성합니다.

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

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

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

1. POD에 볼륨을 마운트합니다.

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

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

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

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

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

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

샘플 매니페스트

PersistentVolumeClaim 샘플 매니페스트

이 예에서는 기본 PVC 구성 옵션을 보여 줍니다.

RWO 액세스가 있는 **PVC**

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

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

NVMe/TCP를 사용한 **PVC**

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

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

Pod 매니페스트 샘플

이 예에서는 PVC를 포드에 연결하는 기본 구성을 보여 줍니다.

기본 구성

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

기본 **NVMe/TCP** 구성

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

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

볼륨 확장

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

iSCSI 볼륨 확장

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

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

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

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

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 `allowVolumeExpansion` 매개 변수를 포함하도록 편집합니다.

2단계: 생성한 StorageClass로 PVC 만들기

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

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

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

Trident는 영구 볼륨(PV)을 생성하고 이 영구 볼륨 클레임(PVC)과 연결합니다.

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

3단계: **PVC**를 연결하는 **POD** 정의

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

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

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

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

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

4단계: **PV** 확장

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

```
kubectl edit pvc san-pvc
```

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

5단계: 확장 유효성 검사

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

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

FC 볼륨 확장

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

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

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

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

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 `allowVolumeExpansion` 매개 변수를 포함하도록 편집합니다.

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

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

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

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

Trident는 영구 볼륨(PV)을 생성하고 이 영구 볼륨 클레임(PVC)과 연결합니다.

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

kubectl get pv
NAME                                       CAPACITY    ACCESS MODES
RECLAIM POLICY    STATUS    CLAIM              STORAGECLASS    REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi         RWO
Delete            Bound     default/san-pvc    ontap-san                 10s
```

3단계: **PVC**를 연결하는 **POD** 정의

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

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

이 예에서는 `san-pvc`를 사용하는 Pod가 생성됩니다.

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

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

4단계: **PV** 확장

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

```
kubectl edit pvc san-pvc
```

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

5단계: 확장 유효성 검사

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

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

NFS 볼륨 확장

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

1단계: 볼륨 확장을 지원하도록 StorageClass 구성

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

```
cat storageclass-ontapnas.yaml
```

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

이 옵션 없이 이미 스토리지 클래스를 생성한 경우 `kubectl edit storageclass`을 사용하여 기존 스토리지 클래스를

편집하여 볼륨 확장을 허용할 수 있습니다.

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

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

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

Trident는 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound     pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

3단계: **PV** 확장

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

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

4단계: 확장 검증

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

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

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
```

볼륨 가져오기

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

개요 및 고려 사항

다음과 같은 경우 Trident로 볼륨을 가져올 수 있습니다.

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

고려 사항

볼륨을 가져오기 전에 다음 사항을 검토하십시오.

- Trident는 RW(읽기-쓰기) 유형의 ONTAP 볼륨만 가져올 수 있습니다. DP(데이터 보호) 유형 볼륨은 SnapMirror 타겟 볼륨입니다. 볼륨을 Trident로 가져오기 전에 미러 관계를 끊어야 합니다.

- 활성 연결이 없는 볼륨을 가져오는 것이 좋습니다. 활성 상태인 볼륨을 가져오려면 볼륨 클론을 생성한 다음 가져오기를 수행하십시오.

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

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

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

- PVC 및 PV에는 볼륨을 가져왔는지 여부와 PVC 및 PV가 관리되는지 여부를 나타내는 이중 목적의 주석이 추가됩니다. 이 주석은 수정하거나 제거해서는 안 됩니다.

볼륨 가져오기

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

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

tridentctl 사용

단계

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

   다음은 최소 사양의 예입니다.

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

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

2. `tridentctl import` 명령을 사용하여 볼륨이 포함된 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume)을 지정합니다. `-f` 인수는 PVC 파일의 경로를 지정하는 데 필요합니다.

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

PVC 주석 사용

단계

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

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

     다음은 관리형 볼륨을 가져오기 위한 예시 사양입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. PVC YAML 파일을 Kubernetes 클러스터에 적용합니다.

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

Trident는 볼륨을 자동으로 가져와 PVC에 바인딩합니다.

예

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하십시오.

ONTAP NAS 및 ONTAP NAS FlexGroup

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

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

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

tridentctl을 사용한 **ONTAP NAS** 예제

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

관리형 볼륨

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

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

관리되지 않는 볼륨

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

다음 예제는 unmanaged_volume`를 `ontap_nas 백엔드에서 가져옵니다:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

PVC 주석을 사용한 **ONTAP NAS** 예제

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

관리형 볼륨

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

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

관리되지 않는 볼륨

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

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

ONTAP SAN

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

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

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리형 볼륨

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

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

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

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

다음 예제는 `unmanaged_example_volume`를 `ontap_san` 백엔드에서 가져옵니다:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

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

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

요소

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

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

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

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

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

Azure NetApp Files

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

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

다음 예제는 백엔드에서 `azure-netapp-files` 볼륨을 가져오며, 볼륨 경로는 `azurenetappfiles_40517`입니다 `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

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

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

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

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

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

볼륨 이름 및 레이블 사용자 지정

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

시작하기 전에

사용자 지정 가능한 볼륨 이름 및 레이블 지원:

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

제한 사항

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

사용자 지정 가능한 볼륨 이름의 주요 동작

- 이름 템플릿의 구문 오류로 인해 오류가 발생하면 백엔드 생성이 실패합니다. 하지만 템플릿 적용이 실패할 경우 볼륨 이름은 기존 명명 규칙에 따라 지정됩니다.

- 백엔드 구성의 이름 템플릿을 사용하여 볼륨 이름을 지정할 때는 스토리지 접두사를 적용할 수 없습니다. 원하는 접두사 값을 템플릿에 직접 추가할 수 있습니다.

이름 템플릿 및 레이블이 포함된 백엔드 구성 예

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

루트 레벨 예

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

풀 수준 예

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

이름 템플릿 예

예 **1**:

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

예 **2**:

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

고려해야 할 사항

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

네임스페이스 간에 **NFS** 볼륨 공유

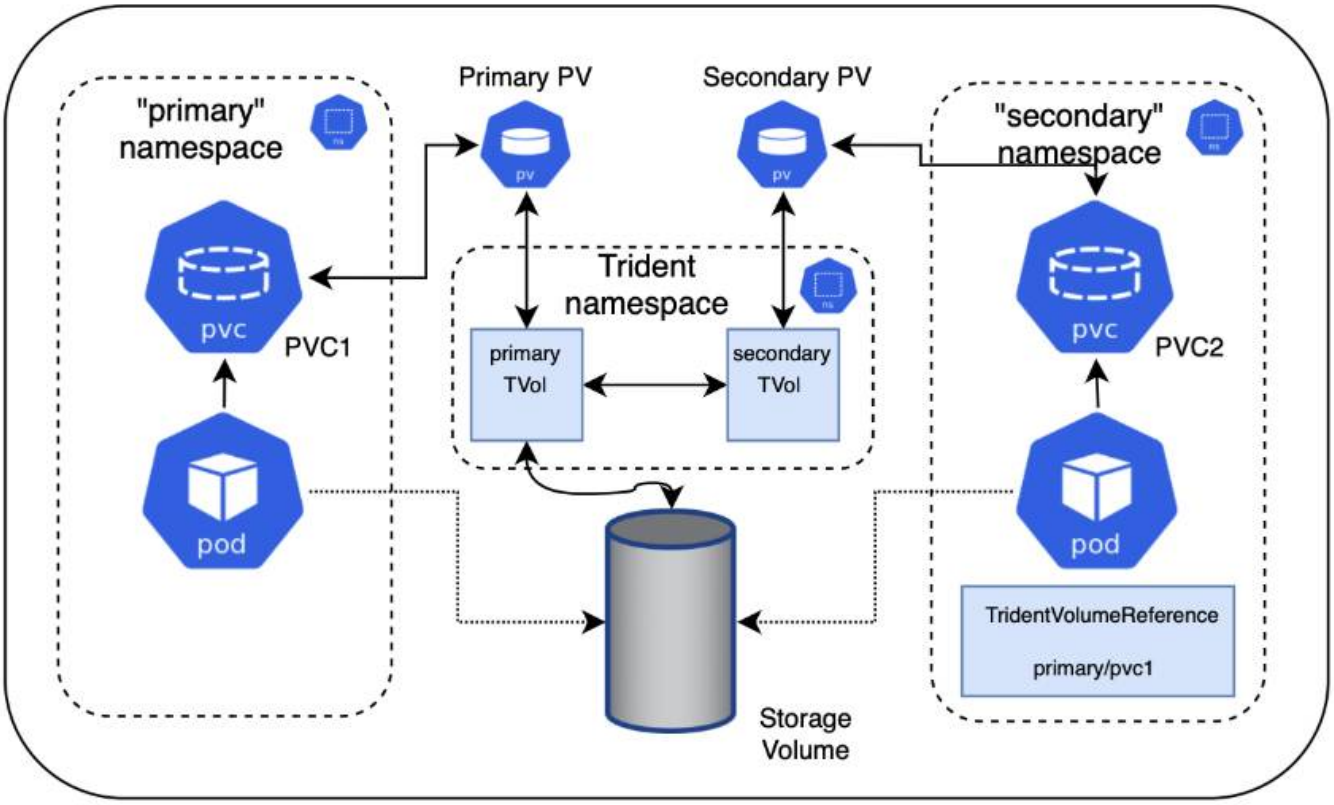
Trident를 사용하면 기본 네임스페이스에 볼륨을 생성하고 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

기능

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

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

이 다이어그램은 두 Kubernetes 네임스페이스에서 NFS 볼륨 공유를 보여줍니다.



빠른 시작

몇 단계만 거치면 NFS 볼륨 공유를 설정할 수 있습니다.

1 소스 **PVC**를 구성하여 볼륨을 공유합니다

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

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

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

3 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4 대상 네임스페이스에 하위 **PVC**를 생성합니다

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

소스 및 대상 네임스페이스를 구성합니다

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

단계

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

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

Trident는 PV와 해당 백엔드 NFS 스토리지 볼륨을 생성합니다.

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

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 TridentVolumeReference CR을 생성합니다 `pvc1`.

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

4. 대상 네임스페이스 소유자: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, 소스 PVC를

지정하기 위해 `shareFromPVC` 주석을 사용합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

ⓘ 대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

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

공유 볼륨 삭제

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

`tridentctl get`를 사용하여 하위 볼륨을 쿼리합니다

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

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

플래그:

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

제한 사항

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

자세한 내용은

교차 네임스페이스 볼륨 액세스에 대한 자세한 내용은 다음을 참조하십시오.

- "네임스페이스 간 볼륨 공유: 네임스페이스 간 볼륨 액세스를 경험해 보세요"를 방문하십시오.
- "NetAppTV"에서 데모를 시청하십시오.

네임스페이스 간 볼륨 복제

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

필수 구성 요소

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

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

빠른 시작

몇 단계만 거치면 볼륨 클로닝을 설정할 수 있습니다.

1. 소스 **PVC**를 구성하여 볼륨 복제

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

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

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

3. 대상 네임스페이스에 **TridentVolumeReference**를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 TridentVolumeReference CR을 생성합니다.

4. 대상 네임스페이스에 클론 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 소스 네임스페이스의 PVC를 복제하기 위해 PVC를 생성합니다.

소스 및 대상 네임스페이스를 구성합니다

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

단계

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

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

Trident가 PV와 해당 백엔드 스토리지 볼륨을 생성합니다.

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

2. 클러스터 관리자: 대상 네임스페이스 소유자가 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있도록 적절한 RBAC가 설정되어 있는지 확인하십시오(`namespace2`.
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 네임스페이스를 참조하는 TridentVolumeReference CR을 생성합니다 `pvc1`.

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

4. 대상 네임스페이스 소유자: 대상 네임스페이스 (`namespace2`)에서 PVC (`pvc2`)를 생성하고, `cloneFromPVC` 또는 `cloneFromSnapshot` 및 `cloneFromNamespace` 주석을 사용하여 소스 PVC를 지정합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

제한 사항

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

SnapMirror를 사용하여 볼륨 복제

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

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

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

복제 사전 요구 사항

시작하기 전에 다음 사전 요구 사항이 충족되었는지 확인하십시오.

ONTAP 클러스터

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

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

  SnapMirror 비동기 보호 기능만 지원됩니다.

피어링

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

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

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

지원되는 드라이버

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

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

미러링된 **PVC** 생성

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

단계

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

예

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

b. 이전에 생성한 StorageClass로 PVC를 생성합니다.

예

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

c. 로컬 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

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

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

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻으십시오.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

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

      예

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

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

예

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

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

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

예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

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

볼륨 복제 상태

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

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

- 타겟 볼륨이 소스 볼륨과 관계를 맺은 적이 있는 경우 타겟 볼륨 콘텐츠를 덮어쓰므로 재설정된 상태를 사용해야 합니다.
- 볼륨이 이전에 소스와 관계를 맺지 않았던 경우 재설정된 상태가 실패합니다.

예기치 않은 페일오버 시 보조 **PVC** 승격

보조 Kubernetes 클러스터에서 다음 단계를 수행하십시오.

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

계획된 페일오버 중에 보조 **PVC**를 승격합니다

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

단계

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

   예

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

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

페일오버 후 미러 관계 복원

미러 관계를 복원하기 전에 새 운영 환경으로 지정할 측을 선택합니다.

단계

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

추가 작업

Trident는 운영 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

기본 **PVC**를 새 보조 **PVC**로 복제합니다

기본 PVC와 보조 PVC가 이미 있는지 확인하십시오.

단계

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

미러링된 운영 또는 보조 **PVC**의 크기 조정

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

PVC에서 복제를 제거합니다

복제를 제거하려면 현재 보조 볼륨에서 다음 작업 중 하나를 수행하십시오.

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

PVC 삭제**(**이전에 미러링됨**)**

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

TMR 삭제

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

ONTAP이 온라인 상태일 때 미러 관계를 업데이트합니다

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

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

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

예

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

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

CSI 토폴로지 사용

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

개요

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

CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

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

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

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- "지원되는 Kubernetes 버전"를 실행하는 Kubernetes 클러스터

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- 클러스터의 노드에는 토폴로지 인식을 도입하는 레이블이 있어야 합니다(`topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`). 이러한 레이블은 Trident가 토폴로지를 인식하도록 하려면 Trident를 설치하기 전에 클러스터의 노드에 있어야 합니다.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

1단계: 토폴로지 인식 백엔드 생성

Trident 스토리지 백엔드는 가용성 영역을 기반으로 볼륨을 선택적으로 프로비저닝하도록 설계할 수 있습니다. 각 백엔드는 지원되는 영역 및 리전 목록을 나타내는 선택적 `supportedTopologies` 블록을 포함할 수 있습니다. 이러한 백엔드를 사용하는 StorageClasses의 경우, 지원되는 리전/영역에 스케줄링된 애플리케이션에서 요청한 경우에만 볼륨이 생성됩니다.

다음은 백엔드 정의의 예입니다.

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

`supportedTopologies`은 백엔드별 리전 및 영역 목록을 제공하는 데 사용됩니다. 이러한 리전 및 영역은 StorageClass에 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에 제공된 리전 및 영역의 하위 집합을 포함하는 StorageClasses의 경우 Trident는 해당 백엔드에 볼륨을 생성합니다.

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

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

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

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

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

다음 예를 참조하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

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

3단계: **PVC** 생성 및 사용

StorageClass를 생성하고 백엔드에 매핑했으면 이제 PVC를 생성할 수 있습니다.

아래 예시를 참조하세요 `spec`:

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

이 매니페스트를 사용하여 PVC를 생성하면 다음과 같은 결과가 나타납니다.

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

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

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

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

다음 출력을 참조하십시오.

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

백엔드를 업데이트하여 다음을 포함하도록 합니다 `supportedTopologies`

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

자세한 정보 찾기

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

스냅샷 작업

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

개요

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

시작하기 전에

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

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

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

볼륨 스냅샷을 생성합니다

단계

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

   예

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

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

2. 기존 PVC의 스냅샷을 생성합니다.

   예
   - 이 예제는 기존 PVC의 스냅샷을 생성합니다.

   ```
   cat snap.yaml
   ```

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

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

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

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

- `VolumeSnapshotContent` 객체를 `pvc1-snap` VolumeSnapshot에 대해 설명하여 식별할 수 있습니다. `Snapshot Content Name`는 이 스냅샷을 제공하는 `VolumeSnapshotContent` 객체를 식별합니다. `Ready To Use` 매개변수는 스냅샷을 사용하여 새 PVC를 생성할 수 있음을 나타냅니다.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

볼륨 스냅샷에서 **PVC**를 생성합니다

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

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

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

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

볼륨 스냅샷 가져오기

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

시작하기 전에

Trident가 스냅샷의 상위 볼륨을 생성하거나 가져와야 합니다.

단계

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

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

     예

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

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

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

   예

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

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

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

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

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

   예

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

스냅샷을 사용하여 볼륨 데이터 복구

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

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

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

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

스냅샷에서 볼륨 제자리 복원

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

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

시작하기 전에

바인딩된 PVC와 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 바인딩되었는지 확인합니다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

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

   TASR CR은 PVC 및 VS가 존재하는 네임스페이스에 있어야 합니다.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 스냅샷에서 복원하려면 CR을 적용하세요. 이 예에서는 스냅샷에서 복원합니다 `pvc1`.

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

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

결과

Trident 스냅샷에서 데이터를 복원합니다. 스냅샷 복원 상태는 다음과 같이 확인할 수 있습니다.

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 대부분의 경우 Trident는 실패 시 자동으로 작업을 재시도하지 않습니다. 작업을 다시 수행해야 합니다.
- 관리자 액세스 권한이 없는 Kubernetes 사용자는 관리자로부터 애플리케이션 네임스페이스에 TASR CR을 생성할 수 있는 권한을 부여받아야 할 수도 있습니다.

연결된 스냅샷이 있는 **PV** 삭제

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

볼륨 스냅샷 컨트롤러를 배포합니다

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

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

ⓘ 필요한 경우 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "볼륨 스냅샷"
- "VolumeSnapshotClass"

볼륨 그룹 스냅샷 작업

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

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

볼륨 그룹 스냅샷 만들기

볼륨 그룹 스냅샷은 다음 스토리지 드라이버에서 지원됩니다.

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

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

시작하기 전에

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

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

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

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

단계

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

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

  다음 예는 `pvc1-group-snap`을 데이터 소스로 사용하고 `consistentGroupSnapshot: groupA`을 레이블로 사용하여 PVC를 생성합니다. 요구 사항에 따라 레이블 키와 값을 정의합니다.

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

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

  이 예에서는 볼륨 그룹 스냅샷을 생성합니다.

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

그룹 스냅샷을 사용하여 볼륨 데이터 복구

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

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

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

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

스냅샷에서 볼륨 제자리 복원

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

자세한 내용은 "스냅샷에서 볼륨 제자리 복원"를 참조하십시오.

연결된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

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

볼륨 스냅샷 컨트롤러를 배포합니다

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

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 스냅샷 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

필요한 경우 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`를 열고 `namespace`를 네임스페이스로 업데이트하십시오.

관련 링크

- "VolumeGroupSnapshotClass"
- "볼륨 스냅샷"

Trident 관리 및 모니터링

Trident 업그레이드

Trident 업그레이드

24.02 릴리스부터 Trident는 4개월 주기로 릴리스를 진행하며, 매년 세 번의 주요 릴리스를 제공합니다. 각 새 릴리스는 이전 릴리스를 기반으로 새로운 기능, 성능 향상, 버그 수정 및 개선 사항을 제공합니다. Trident의 새로운 기능을 활용하려면 1년에 한 번 이상 업그레이드하는 것이 좋습니다.

업그레이드 전 고려 사항

Trident의 최신 릴리스로 업그레이드할 때 다음 사항을 고려하십시오.

- 주어진 Kubernetes 클러스터 내의 모든 네임스페이스에는 하나의 Trident 인스턴스만 설치되어야 합니다.
- Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 알파 또는 베타 스냅샷은 더 이상 지원되지 않습니다.
- 업그레이드할 때, Trident에서 사용하는 `parameter.fsType`에 `StorageClasses`를 제공하는 것이 중요합니다. 기존 볼륨에 영향을 주지 않고 `StorageClasses`를 삭제하고 다시 생성할 수 있습니다.
  - 이는 SAN 볼륨에 대한 "보안 컨텍스트" 적용을 위한 필수 요건입니다.
  - 샘플 입력 디렉토리에는 `storage-class-basic.yaml.templ` 및 `storage-class-bronze-default.yaml`와 같은 예제가 포함되어 있습니다.
  - 자세한 내용은 "알려진 문제"을(를) 참조하십시오.

1단계**:** 버전 선택

Trident 버전은 날짜 기반 `YY.MM` 명명 규칙을 따르며, 여기서 "YY"는 연도의 마지막 두 자리 숫자이고 "MM"은 월을 나타냅니다. Dot 릴리스는 `YY.MM.X` 규칙을 따르며, 여기서 "X"는 패치 레벨을 나타냅니다. 업그레이드할 버전은 업그레이드하려는 현재 버전에 따라 선택합니다.

- 설치된 버전에서 4개 릴리스 이내의 대상 릴리스로 직접 업그레이드할 수 있습니다. 예를 들어 24.06(또는 24.06의 모든 마이너 버전)에서 25.06으로 직접 업그레이드할 수 있습니다.
- 4개 릴리스 기간 외의 버전에서 업그레이드하는 경우 여러 단계를 거쳐 업그레이드하십시오. "이전 버전"에서 업그레이드하는 경우 해당 버전의 업그레이드 지침을 사용하여 4개 릴리스 기간에 맞는 최신 릴리스로 업그레이드하십시오. 예를 들어 23.07을 실행 중이고 25.06으로 업그레이드하려는 경우:
  a. 먼저 23.07에서 24.06으로 업그레이드하십시오.
  b. 그다음 24.06에서 25.06으로 업그레이드하세요.

> OpenShift Container Platform에서 Trident 오퍼레이터를 사용하여 업그레이드할 경우 Trident 21.01.1 이상으로 업그레이드해야 합니다. 21.01.0과 함께 릴리스된 Trident 오퍼레이터에는 알려진 문제가 있으며 이 문제는 21.01.1에서 수정되었습니다. 자세한 내용은 "GitHub의 문제 세부 정보"을 참조하십시오.

2단계**:** 원래 설치 방법을 확인합니다

Trident를 원래 설치하는 데 사용한 버전을 확인하려면:

1. `kubectl get pods -n trident`를 사용하여 Pod를 검사합니다.
   - 오퍼레이터 포드가 없는 경우 Trident가 `tridentctl`을 사용하여 설치되었습니다.
   - 오퍼레이터 포드가 있는 경우, Trident는 수동으로 또는 Helm을 사용하여 Trident 오퍼레이터를 통해 설치되었습니다.
2. 운영자 포드가 있는 경우 `kubectl describe torc`를 사용하여 Helm을 통해 Trident가 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Trident는 Helm을 사용하여 설치되었습니다.
   - Helm 레이블이 없으면 Trident 운영자를 사용하여 Trident를 수동으로 설치한 것입니다.

3단계**:** 업그레이드 방법 선택

일반적으로 초기 설치에 사용했던 것과 동일한 방법으로 업그레이드하는 것이 좋지만, "설치 방법 간 이동". Trident를 업그레이드하는 두 가지 옵션이 있습니다.

- "Trident 운영자를 사용하여 업그레이드하세요"

> 운영자와 함께 업그레이드하기 전에 "운영자 업그레이드 워크플로 이해"을(를) 검토하는 것이 좋습니다.

*

운영자를 통해 업그레이드

운영자 업그레이드 워크플로 이해

Trident 오퍼레이터를 사용하여 Trident를 업그레이드하기 전에 업그레이드 중에 발생하는 백그라운드 프로세스를 이해해야 합니다. 여기에는 롤링 업데이트를 가능하게 하는 Trident 컨트롤러, 컨트롤러 Pod 및 노드 Pod, 노드 DaemonSet의 변경 사항이 포함됩니다.

Trident 운영자 업그레이드 처리

Trident를 설치 및 업그레이드하는 많은 "Trident 운영자 사용의 이점" 중 하나는 기존에 마운트된 볼륨에 영향을 주지 않고 Trident 및 Kubernetes 객체를 자동으로 처리한다는 점입니다. 이러한 방식으로 Trident는 다운타임 없이 업그레이드를 지원할 수 있습니다("롤링 업데이트"). 특히 Trident 오퍼레이터는 Kubernetes 클러스터와 통신하여 다음을 수행합니다.

- Trident Controller 배포 및 노드 DaemonSet을 삭제하고 다시 생성합니다.
- Trident Controller Pod와 Trident Node Pod를 새 버전으로 교체하십시오.
  - 노드가 업데이트되지 않더라도 나머지 노드의 업데이트가 차단되지 않습니다.
  - 실행 중인 Trident Node Pod가 있는 노드만 볼륨을 마운트할 수 있습니다.

> Kubernetes 클러스터의 Trident 아키텍처에 대한 자세한 내용은 "Trident 아키텍처"을 참조하십시오.

Operator 업그레이드 워크플로

Trident 운영자를 사용하여 업그레이드를 시작할 때:

1. **Trident** 운영자:
   a. 현재 설치된 Trident 버전(*n*)을 감지합니다.
   b. CRD, RBAC 및 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** 운영자:
   a. _n_에 대한 Trident 노드 DaemonSet을 삭제합니다. 운영자는 노드 Pod 종료를 기다리지 않습니다.
   b. _n+1_에 대한 Trident 노드 데몬셋을 생성합니다.
4. *Kubernetes*는 Trident Node Pod _n_이 실행 중이지 않은 노드에 Trident Node Pod을 생성합니다. 이를 통해 노드에는 어떤 버전이든 Trident Node Pod이 두 개 이상 존재하지 않게 됩니다.

Trident Operator 또는 Helm을 사용하여 Trident 설치를 업그레이드하세요

Trident 운영자를 사용하여 수동으로 또는 Helm을 통해 Trident를 업그레이드할 수 있습니다. Trident 운영자 설치에서 다른 Trident 운영자 설치로 업그레이드하거나 `tridentctl` 설치에서 Trident 운영자 버전으로 업그레이드할 수 있습니다. Trident 운영자 설치를 업그레이드하기 전에 "업그레이드 방법을 선택하세요"를 검토하십시오.

수동 설치 업그레이드

클러스터 범위 Trident 운영자 설치에서 다른 클러스터 범위 Trident 운영자 설치로 업그레이드할 수 있습니다. 모든 Trident 버전은 클러스터 범위 운영자를 사용합니다.

> ⓘ 네임스페이스 범위 연산자를 사용하여 설치된 Trident(버전 20.07~20.10)에서 업그레이드하려면 "설치된 버전" Trident의 업그레이드 지침을 사용하십시오.

이 작업 정보

Trident는 Kubernetes 버전에 맞는 오퍼레이터를 설치하고 관련 객체를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.

- Kubernetes 1.24를 실행하는 클러스터의 경우 "bundle_pre_1_25.yaml"을(를) 사용하십시오.
- Kubernetes 1.25 이상 버전을 실행하는 클러스터의 경우 "bundle_post_1_25.yaml"를 사용하십시오.

시작하기 전에

"지원되는 Kubernetes 버전"을(를) 실행하는 Kubernetes 클러스터를 사용하고 있는지 확인하십시오.

단계

1. Trident 버전을 확인하세요:

```
./tridentctl -n trident version
```

2. `operator.yaml`, `tridentorchestrator_cr.yaml`, `post_1_25_bundle.yaml`를 업그레이드할 버전(예: 25.06)에 맞는 레지스트리 및 이미지 경로와 올바른 시크릿으로 업데이트하십시오.
3. 현재 Trident 인스턴스를 설치하는 데 사용된 Trident 오퍼레이터를 삭제하십시오. 예를 들어 25.02에서 업그레이드하는 경우 다음 명령을 실행하십시오.

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. `TridentOrchestrator` 속성을 사용하여 초기 설치를 사용자 지정한 경우 `TridentOrchestrator` 객체를 편집하여 설치 매개 변수를 수정할 수 있습니다. 여기에는 오프라인 모드용 미러링된 Trident 및 CSI 이미지 레지스트리 지정, 디버그 로그 활성화 또는 이미지 풀 시크릿 지정을 위한 변경 사항이 포함될 수 있습니다.
5. Kubernetes 버전에 따라 _<bundle.yaml>_가 `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml`인 환경에 맞는 번들 YAML 파일을 사용하여 Trident를 설치합니다. 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. Trident torc를 수정하여 이미지 25.06.0을 포함시키세요.

Helm 설치 업그레이드

Trident Helm 설치를 업그레이드할 수 있습니다.

> Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상 버전으로 업그레이드할 때는 클러스터를 업그레이드하기 전에 values.yaml을 업데이트하여 `excludePodSecurityPolicy`을 `true`로 설정하거나 `--set excludePodSecurityPolicy=true`을 `helm upgrade` 명령에 추가해야 합니다.

Kubernetes 클러스터를 1.24에서 1.25로 업그레이드했지만 Trident helm을 업그레이드하지 않은 경우 helm 업그레이드가 실패합니다. helm 업그레이드를 성공적으로 완료하려면 다음 단계를 사전 수행해야 합니다.

1. https://github.com/helm/helm-mapkubeapis에서 helm-mapkubeapis 플러그인을 설치합니다.
2. Trident가 설치된 네임스페이스에서 Trident 릴리스에 대한 사전 실행(dry run)을 수행합니다. 이렇게 하면 정리될 리소스 목록이 표시됩니다.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. 정리 작업을 위해 helm을 사용하여 전체 실행을 수행하십시오.

```
helm mapkubeapis trident  --namespace trident
```

단계

1. "Helm을 사용하여 Trident를 설치했습니다"한 경우 `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0`를 사용하여 한 단계로 업그레이드할 수 있습니다. Helm 리포지토리를 추가하지 않았거나 업그레이드에 사용할 수 없는 경우:

   a. 최신 Trident 릴리스를 "GitHub의 *Assets* 섹션"에서 다운로드하십시오.

   b. `helm upgrade` 명령을 사용합니다. 여기서 `trident-operator-25.10.0.tgz`는 업그레이드할 버전을 나타냅니다.

   ```
   helm upgrade <name> trident-operator-25.10.0.tgz
   ```

   초기 설치 중에 사용자 지정 옵션(예: Trident 및 CSI 이미지에 대한 개인 미러링 레지스트리 지정)을 설정한 경우 `helm upgrade` 명령을 `--set`를 사용하여 추가하여 해당 옵션이 업그레이드 명령에 포함되도록 해야 합니다. 그렇지 않으면 값이 기본값으로 재설정됩니다.

2. `helm list`을 실행하여 차트와 앱 버전이 모두 업그레이드되었는지 확인합니다. `tridentctl logs`를 실행하여 디버그 메시지를 검토합니다.

`tridentctl` 설치에서 **Trident operator**로 업그레이드

`tridentctl` 설치에서 `Trident` 운영자의 최신 릴리스로 업그레이드할 수 있습니다. 기존 백엔드와 `PVC`는 자동으로 사용할 수 있습니다.

설치 방법을 전환하기 전에 "설치 방법 간 이동"을(를) 검토하십시오.

단계

1. 최신 Trident 릴리스를 다운로드하세요.

   ```
   # Download the release required [25.10.0]
   mkdir 25.10.0
   cd 25.10.0
   wget
   https://github.com/NetApp/trident/releases/download/v25.10.0/trident-
   installer-25.10.0.tar.gz
   tar -xf trident-installer-25.10.0.tar.gz
   cd trident-installer
   ```

2. 매니페스트에서 `tridentorchestrator` CRD를 생성합니다.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 클러스터 범위 오퍼레이터를 동일한 네임스페이스에 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. Trident 설치를 위한 `TridentOrchestrator` CR을 생성합니다.

```
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. Trident가 의도한 버전으로 업그레이드되었는지 확인하십시오.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.10.0
```

tridentctl로 업그레이드합니다

`tridentctl`를 사용하여 기존 Trident 설치를 쉽게 업그레이드할 수 있습니다.

이 작업 정보

Trident를 제거했다가 다시 설치하는 것은 업그레이드와 같습니다. Trident를 제거해도 Trident 배포에서 사용되는 영구 볼륨 클레임(PVC) 및 영구 볼륨(PV)은 삭제되지 않습니다. 이미 프로비저닝된 PV는 Trident가 오프라인 상태인 동안에도 계속 사용 가능하며, Trident는 온라인 상태로 돌아온 후 그 사이에 생성된 모든 PVC에 대해 볼륨을 프로비저닝합니다.

시작하기 전에

업그레이드하기 전에 "업그레이드 방법을 선택하세요"를 검토한 후 `tridentctl`를 사용하십시오.

단계

1. `tridentctl`에서 제거 명령을 실행하여 CRD 및 관련 개체를 제외한 Trident와 관련된 모든 리소스를 제거합니다.

```
./tridentctl uninstall -n <namespace>
```

2. Trident를 다시 설치합니다. 다음을 참조하십시오."tridentctl을 사용하여 Trident를 설치합니다"

업그레이드 프로세스를 중단하지 마십시오. 설치 프로그램이 완료될 때까지 실행되는지 확인하십시오.

tridentctl을 사용하여 Trident를 관리합니다

https://github.com/NetApp/trident/releases["Trident 설치 프로그램 번들"^]에는 Trident에 대한 간단한 액세스를 제공하는 `tridentctl` 명령줄 유틸리티가 포함되어 있습니다. 충분한 권한을 가진 Kubernetes 사용자는 이 유틸리티를 사용하여 Trident를 설치하거나 Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

명령 및 글로벌 플래그

`tridentctl help`을 실행하여 `tridentctl`에 사용 가능한 명령 목록을 가져오거나 `--help` 플래그를 명령에 추가하여 해당 특정 명령에 대한 옵션 및 플래그 목록을 가져올 수 있습니다.

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl` 유틸리티는 다음과 같은 명령과 전역 플래그를 지원합니다.

명령

create
Trident에 리소스를 추가합니다.

delete
Trident에서 하나 이상의 리소스를 제거합니다.

get
Trident에서 하나 이상의 리소스를 가져옵니다.

help
모든 명령에 대한 도움말입니다.

images
Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

import
기존 리소스를 Trident로 가져옵니다.

install
Trident를 설치합니다.

logs
Trident에서 로그를 인쇄합니다.

send
Trident에서 리소스를 보냅니다.

uninstall
Trident를 제거합니다.

update
Trident에서 리소스를 수정합니다.

update backend state
백엔드 작업을 일시적으로 일시 중단합니다.

upgrade
Trident에서 리소스를 업그레이드합니다.

version
Trident 버전을 인쇄합니다.

글로벌 플래그

-d, --debug

디버그 출력.

-h, --help

`tridentctl`에 대한 도움말.

-k, --kubeconfig string

`KUBECONFIG` 경로를 지정하여 로컬에서 또는 한 Kubernetes 클러스터에서 다른 클러스터로 명령을 실행합니다.

또는 `KUBECONFIG` 변수를 내보내 특정 Kubernetes 클러스터를 가리키도록 한 다음 해당 클러스터에 `tridentctl` 명령을 실행할 수도 있습니다.

-n, --namespace string

Trident 배포의 네임스페이스입니다.

-o, --output string

출력 형식. json|yaml|name|wide|ps 중 하나입니다(기본값).

-s, --server string

Trident REST 인터페이스의 주소/포트.

Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 대기하고 서비스를 제공하도록 구성할 수 있습니다.

명령 옵션 및 플래그

생성

`create` 명령을 사용하여 `Trident`에 리소스를 추가합니다.

```
tridentctl create [option]
```

옵션

`backend`: Trident에 백엔드를 추가합니다.

삭제

`delete` 명령을 사용하여 `Trident`에서 하나 이상의 리소스를 제거합니다.

```
tridentctl delete [option]
```

옵션

backend: Trident에서 하나 이상의 스토리지 백엔드를 삭제합니다.
snapshot: Trident에서 하나 이상의 볼륨 스냅샷을 삭제합니다.
storageclass: Trident에서 하나 이상의 스토리지 클래스를 삭제합니다.
volume: Trident에서 하나 이상의 스토리지 볼륨을 삭제합니다.

가져오기

`get` 명령을 사용하여 Trident에서 하나 이상의 리소스를 가져옵니다.

```
tridentctl get [option]
```

옵션

backend: Trident에서 하나 이상의 스토리지 백엔드를 가져옵니다.
snapshot: Trident에서 하나 이상의 스냅샷을 가져옵니다.
storageclass: Trident에서 하나 이상의 스토리지 클래스를 가져옵니다.
volume: Trident에서 하나 이상의 볼륨을 가져옵니다.

플래그

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

이미지

`images` 플래그를 사용하여 Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

```
tridentctl images [flags]
```

플래그

-h, --help: 이미지 관련 도움말입니다.
-v, --k8s-version string: Kubernetes 클러스터의 시맨틱 버전입니다.

볼륨 가져오기

`import volume` 명령을 사용하여 기존 볼륨을 Trident로 가져옵니다.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

별칭

volume, v

플래그

-f, --filename string: YAML 또는 JSON PVC 파일 경로.
-h, --help: 볼륨에 대한 도움말.

`--no-manage`: PV/PVC만 생성합니다. 볼륨 수명 주기 관리는 고려하지 않습니다.

설치

`install` 플래그를 사용하여 `Trident`를 설치합니다.

```
tridentctl install [flags]
```

플래그

`--autosupport-image string`: Autosupport Telemetry용 컨테이너 이미지(기본값 "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: Autosupport Telemetry 전송을 위한 프록시 주소/포트.
`--enable-node-prep`: 노드에 필요한 패키지 설치를 시도합니다.
`--generate-custom-yaml`: 아무것도 설치하지 않고 YAML 파일을 생성합니다.
`-h, --help`: 설치에 대한 도움말.
`--http-request-timeout`: Trident 컨트롤러 REST API의 HTTP 요청 시간 초과를 재정의합니다(기본값 1m30s).
`--image-registry string`: 내부 이미지 레지스트리의 주소/포트.
`--k8s-timeout duration`: 모든 Kubernetes 작업의 시간 초과(기본값 3m0s).
`--kubelet-dir string`: kubelet의 내부 상태 호스트 위치(기본값 "/var/lib/kubelet").
`--log-format string`: Trident 로깅 형식(text, json)(기본값 "text").
`--node-prep`: 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 Trident를 활성화합니다. 현재 **`iscsi`만 지원됩니다. `OpenShift 4.19`부터 이 기능을 지원하는 최소 `Trident` 버전은 `25.06.1`입니다.**
`--pv string: Trident에서 사용하는 레거시 PV의 이름입니다. 이 이름이 존재하지 않도록 합니다(기본값 "trident").
`--pvc string`: Trident에서 사용하는 레거시 PVC의 이름입니다. 이 이름이 존재하지 않도록 합니다(기본값 "trident").
`--silence-autosupport`: autosupport 번들을 NetApp에 자동으로 보내지 않습니다(기본값 true).
`--silent`: 설치 중 대부분의 출력을 비활성화합니다.
`--trident-image string`: 설치할 Trident 이미지입니다.
`--k8s-api-qps`: Kubernetes API 요청에 대한 초당 쿼리 수(QPS) 제한입니다(기본값 100, 선택 사항).
`--use-custom-yaml`: 설정 디렉터리에 있는 기존 YAML 파일을 사용합니다.
`--use-ipv6`: Trident 통신에 IPv6를 사용합니다.

로그

`logs` 플래그를 사용하여 `Trident`의 로그를 인쇄합니다.

```
tridentctl logs [flags]
```

플래그

`-a, --archive`: 별도로 지정하지 않는 한 모든 로그를 포함하는 지원 아카이브를 생성합니다.
`-h, --help`: 로그에 대한 도움말입니다.
`-l, --log string`: 표시할 Trident 로그입니다. trident|auto|trident-operator|all 중 하나입니다(기본값 "auto").
`--node string`: 노드 Pod 로그를 수집할 Kubernetes 노드 이름입니다.
`-p, --previous`: 이전 컨테이너 인스턴스가 있는 경우 해당 인스턴스의 로그를 가져옵니다.
`--sidecars`: 사이드카 컨테이너의 로그를 가져옵니다.

전송

`send` 명령을 사용하여 Trident에서 리소스를 전송합니다.

```
tridentctl send [option]
```

옵션

autosupport: NetApp에 Autosupport 아카이브를 보냅니다.

제거

`uninstall` 플래그를 사용하여 Trident를 제거합니다.

```
tridentctl uninstall [flags]
```

플래그

-h, --help: 제거 관련 도움말입니다.
--silent: 제거 과정에서 대부분의 출력을 비활성화합니다.

업데이트

`update` 명령을 사용하여 Trident에서 리소스를 수정합니다.

```
tridentctl update [option]
```

옵션

backend: Trident에서 백엔드를 업데이트합니다.

백엔드 상태 업데이트

`update backend state` 명령을 사용하여 백엔드 작업을 일시 중단하거나 재개합니다.

```
tridentctl update backend state <backend-name> [flag]
```

고려해야 할 사항

- TridentBackendConfig(tbc)를 사용하여 백엔드를 생성한 경우 backend.json 파일을 사용하여 백엔드를 업데이트할 수 없습니다.
- userState`이 tbc에 설정된 경우 `tridentctl update backend state <backend-name> --user-state suspended/normal 명령을 사용하여 수정할 수 없습니다.
- tbc를 통해 설정한 후 tridentctl을 통해 userState`을(를) 다시 설정할 수 있도록 하려면 `userState 필드를 tbc에서 제거해야 합니다. 이는 kubectl edit tbc 명령을 사용하여 수행할 수 있습니다. userState 필드를 제거한 후에는 tridentctl update backend state 명령을 사용하여 백엔드의 `userState`을(를) 변경할 수 있습니다.

- `tridentctl update backend state`를 사용하여 `userState`를 변경합니다. 또한 `userState`를 `TridentBackendConfig` 또는 `backend.json` 파일을 사용하여 업데이트할 수 있습니다. 이 작업은 백엔드의 전체 재초기화를 트리거하며 시간이 오래 걸릴 수 있습니다.

  플래그

  `-h`, `--help`: 백엔드 상태에 대한 도움말입니다.
  `--user-state`: 백엔드 작업을 일시 중지하려면 `suspended`로 설정하십시오. 백엔드 작업을 재개하려면 `normal`로 설정하십시오. `suspended`로 설정하면:

- `AddVolume` 및 `Import Volume`이(가) 일시 중지됩니다.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess`을(를) 계속 사용할 수 있습니다.

> `userState` 필드를 사용하여 백엔드 구성 파일 `TridentBackendConfig` 또는 `backend.json`에서 백엔드 상태를 업데이트할 수도 있습니다. 자세한 내용은 xref:{relative_path}../trident-use/backend_options.html["백엔드 관리 옵션"] 및 xref:{relative_path}../trident-use/backend_ops_kubectl.html["kubectl을 사용하여 백엔드 관리를 수행합니다"]을 참조하십시오.

예:

JSON

다음 단계를 따라 `userState`를 `backend.json` 파일을 사용하여 업데이트하십시오:

1. `backend.json` 파일을 편집하여 `userState` 필드를 값이 'suspended'로 설정된 상태로 포함하십시오.
2. `tridentctl update backend` 명령어와 업데이트된 `backend.json` 파일의 경로를 사용하여 백엔드를 업데이트하세요.

   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

`kubectl edit <tbc-name> -n <namespace>` 명령을 사용하여 적용된 후 `tbc`를 편집할 수 있습니다. 다음 예에서는 `userState: suspended` 옵션을 사용하여 백엔드 상태를 일시 중단으로 업데이트합니다.

```
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

`version` 플래그를 사용하여 `tridentctl` 및 실행 중인 Trident 서비스의 버전을 출력합니다.

```
tridentctl version [flags]
```

플래그

--client: 클라이언트 버전만 해당됩니다(서버 필요 없음).
-h, --help: 버전 관련 도움말.

플러그인 지원

Tridentctl은 kubectl과 유사한 플러그인을 지원합니다. Tridentctl은 플러그인 바이너리 파일 이름이 "tridentctl-<plugin>" 체계를 따르고 바이너리가 PATH 환경 변수에 나열된 폴더에 있는 경우 플러그인을 감지합니다. 감지된 모든 플러그인은 tridentctl help의 플러그인 섹션에 나열됩니다. 선택적으로 환경 변수 TRIDENTCTL_PLUGIN_PATH에서 플러그인 폴더를 지정하여 검색을 제한할 수도 있습니다(예: TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/). 변수를 사용하는 경우 tridentctl은 지정된 폴더에서만 검색합니다.

Trident 모니터링

Trident는 Trident 성능을 모니터링하는 데 사용할 수 있는 Prometheus 메트릭 엔드포인트 세트를 제공합니다.

개요

Trident에서 제공하는 메트릭을 통해 다음과 같은 작업을 수행할 수 있습니다.

- Trident의 상태와 구성을 추적합니다. 작업이 얼마나 성공적인지, 그리고 예상대로 백엔드와 통신할 수 있는지 확인할 수 있습니다.
- 백엔드 사용 정보를 분석하여 백엔드에 프로비저닝된 볼륨 수와 사용된 공간의 양 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 수에 대한 매핑을 유지 관리합니다.
- 성능을 추적하세요. Trident가 백엔드와 통신하고 작업을 수행하는 데 걸리는 시간을 확인할 수 있습니다.

기본적으로 Trident의 메트릭은 대상 포트 8001`의 `/metrics 엔드포인트에 노출됩니다. 이러한 메트릭은 Trident 설치 시 *기본적으로 활성화*됩니다. 포트 `8444`에서 HTTPS를 통해 Trident 메트릭을 사용하도록 구성할 수도 있습니다.

필요한 것

- Trident가 설치된 Kubernetes 클러스터.
- Prometheus 인스턴스. 이것은 "컨테이너화된 Prometheus 배포"일 수 있으며, Prometheus를 "네이티브 애플리케이션"로 실행하도록 선택할 수도 있습니다.

1단계: Prometheus 타겟 정의

Trident가 관리하는 백엔드, 생성하는 볼륨 등에 대한 메트릭과 정보를 수집하려면 Prometheus 타겟을 정의해야 합니다. "Prometheus Operator 설명서"을 참조하십시오.

2단계: Prometheus ServiceMonitor 생성

Trident 메트릭을 사용하려면 `trident-csi` 서비스를 감시하고 `metrics` 포트에서 수신 대기하는 Prometheus ServiceMonitor를 생성해야 합니다. 샘플 ServiceMonitor는 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 `trident-csi` 서비스에서 반환되는 메트릭을 검색하고 특히 서비스의 `metrics` 엔드포인트를 찾습니다. 결과적으로 이제 Prometheus는 Trident의 메트릭을 이해하도록 구성되었습니다.

Trident에서 직접 사용할 수 있는 메트릭 외에도 kubelet은 `kubelet_volume_*` 자체 메트릭 엔드포인트를 통해 다양한 메트릭을 제공합니다. Kubelet은 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. "여기"을 참조하십시오.

HTTPS를 통해 Trident 메트릭을 사용합니다

HTTPS(포트 8444)를 통해 Trident 메트릭을 사용하려면, ServiceMonitor 정의를 수정하여 TLS 구성을 포함해야 합니다. 또한 `trident-csi` 시크릿을 `trident` 네임스페이스에서 Prometheus가 실행 중인 네임스페이스로 복사해야 합니다. 다음 명령어를 사용하여 이 작업을 수행할 수 있습니다:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

HTTPS 메트릭을 위한 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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident는 tridentctl, Helm 차트 및 Operator를 포함한 모든 설치 방법에서 HTTPS 메트릭을 지원합니다.

- `tridentctl install` 명령을 사용하는 경우 `--https-metrics` 플래그를 전달하여 HTTPS 메트릭을 활성화할 수 있습니다.
- Helm 차트를 사용하는 경우 `httpsMetrics` 매개변수를 설정하여 HTTPS 메트릭을 활성화할 수 있습니다.
- YAML 파일을 사용하는 경우, `--https_metrics` 플래그를 `trident-main` 컨테이너에 `trident-deployment.yaml` 파일에서 추가할 수 있습니다.

3단계: PromQL을 사용하여 Trident 메트릭을 쿼리합니다

PromQL은 시계열 또는 표 형식 데이터를 반환하는 표현식을 생성하는 데 유용합니다.

다음은 사용할 수 있는 PromQL 쿼리입니다.

Trident 상태 정보를 확인하십시오

- **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
```

- 상태 코드별 **Trident**의 **REST** 응답 비율

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Trident**에서 수행한 작업의 평균 기간**(ms)**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Trident 사용 정보를 확인하세요

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝한 총 볼륨 공간

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

개별 볼륨 사용량 확인

> 이 기능은 kubelet 메트릭도 수집되는 경우에만 활성화됩니다.

- 각 볼륨의 사용된 공간 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Trident AutoSupport 원격 측정에 대해 알아보세요

기본적으로 Trident는 Prometheus 메트릭과 기본 백엔드 정보를 매일 NetApp으로 전송합니다.

- Trident가 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp으로 전송하지 못하도록 하려면 Trident 설치 중에 `--silence-autosupport` 플래그를 전달하십시오.
- Trident는 `tridentctl send autosupport`를 통해 필요에 따라 컨테이너 로그를 NetApp 지원팀에 전송할 수도 있습니다. Trident가 로그를 업로드하도록 트리거해야 합니다. 로그를 제출하기 전에 NetApp 의https://www.netapp.com/company/legal/privacy-policy/["개인정보 보호정책"^]에 동의해야 합니다.
- 별도로 지정하지 않으면 Trident는 지난 24시간 동안의 로그를 가져옵니다.
- `--since` 플래그를 사용하여 로그 보존 기간을 지정할 수 있습니다. 예를 들면 다음과 같습니다 `tridentctl send autosupport --since=1h`. 이 정보는 Trident와 함께 설치되는 `trident-autosupport` 컨테이너를 통해 수집되고 전송됩니다. 컨테이너 이미지는 "Trident AutoSupport"에서 얻을 수 있습니다.
- Trident AutoSupport는 개인 식별 정보(PII) 또는 개인 정보를 수집하거나 전송하지 않습니다. Trident 컨테이너 이미지 자체에는 적용되지 않는 "EULA"와 함께 제공됩니다. NetApp의 데이터 보안 및 신뢰에 대한 노력에 대해 자세히 알아볼 수 있습니다 "여기".

Trident에서 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 저장하는 데 개인 레지스트리를 사용하는 경우 `--image-registry` 플래그를 사용할 수 있습니다.
- 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이 작업은 YAML 파일을 생성하기 위해 `tridentctl install --generate-custom-yaml`를 사용하고, `trident-autosupport` 컨테이너에 `--proxy-url` 인수를 `trident-deployment.yaml`에 추가하여 수행할 수 있습니다.

Trident 메트릭을 비활성화합니다

메트릭 보고를 비활성화하려면, 사용자 지정 YAML을 생성해야 합니다(`--generate-custom-yaml` 플래그 사용). 그런 다음 해당 파일을 편집하여 `--metrics` 플래그가 `trident-main` 컨테이너에서 호출되지 않도록 제거해야 합니다.

Trident 제거

Trident를 설치하는 데 사용한 것과 동일한 방법을 사용하여 Trident를 제거해야 합니다.

이 작업 정보

- 업그레이드 후 발견된 버그, 종속성 문제 또는 업그레이드 실패나 불완전한 업그레이드에 대한 수정이 필요한 경우, Trident를 제거하고 해당 "버전"에 대한 특정 지침을 사용하여 이전 버전을 다시 설치해야 합니다. 이것이 이전 버전으로 다운그레이드 하는 유일하게 권장되는 방법입니다.
- 간편한 업그레이드 및 재설치를 위해 Trident를 제거해도 Trident에서 생성한 CRD 또는 관련 객체는 제거되지 않습니다. Trident와 모든 데이터를 완전히 제거해야 하는 경우 "Trident 및 CRD를 완전히 제거합니다"를 참조하십시오.

시작하기 전에

Kubernetes 클러스터를 사용 중지하는 경우 제거하기 전에 Trident에서 생성한 볼륨을 사용하는 모든 애플리케이션을 삭제해야 합니다. 이렇게 하면 PVC가 삭제되기 전에 Kubernetes 노드에서 게시 취소됩니다.

원래 설치 방법을 확인하십시오

Trident를 설치할 때 사용했던 방법과 동일한 방법으로 Trident를 제거해야 합니다. 제거하기 전에 원래 Trident를 설치하는 데 사용한 버전을 확인하십시오.

1. `kubectl get pods -n trident`를 사용하여 Pod를 검사합니다.
   - 오퍼레이터 포드가 없는 경우 Trident가 `tridentctl`을 사용하여 설치되었습니다.
   - 오퍼레이터 포드가 있는 경우, Trident는 수동으로 또는 Helm을 사용하여 Trident 오퍼레이터를 통해 설치되었습니다.
2. 운영자 포드가 있는 경우 `kubectl describe tproc trident`를 사용하여 Helm을 통해 Trident가 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Trident는 Helm을 사용하여 설치되었습니다.
   - Helm 레이블이 없으면 Trident 운영자를 사용하여 Trident를 수동으로 설치한 것입니다.

Trident 운영자 설치를 제거합니다

Trident Operator 설치는 수동으로 또는 Helm을 사용하여 제거할 수 있습니다.

수동 설치 제거

운영자를 사용하여 Trident를 설치한 경우 다음 중 하나를 수행하여 제거할 수 있습니다.

1. **CR `TridentOrchestrator`**을 편집하고 제거 플래그를 설정하세요:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"uninstall":true}}'
```

`uninstall` 플래그가 `true`로 설정되면 Trident 운영자는 Trident를 제거하지만 TridentOrchestrator 자체는 제거하지 않습니다. Trident를 다시 설치하려면 TridentOrchestrator를 정리하고 새로 생성해야 합니다.

2. 삭제 **TridentOrchestrator**: Trident를 배포하는 데 사용된 TridentOrchestrator CR을 제거하면 운영자에게 Trident를 제거하도록 지시합니다. 운영자는 TridentOrchestrator 제거를 처리하고 Trident 배포 및 데몬셋을 제거하며, 설치 과정에서 생성한 Trident Pod를 삭제합니다.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Helm 설치 제거

Helm을 사용하여 Trident를 설치한 경우 `helm uninstall`를 사용하여 제거할 수 있습니다.

```
#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
```

tridentctl 설치 제거

Trident와 관련된 CRD 및 관련 오브젝트를 제외한 모든 리소스를 삭제하려면 uninstall 명령을 `tridentctl`에서 사용하십시오.

```
./tridentctl uninstall -n <namespace>
```

Docker용 Trident

배포를 위한 사전 요구 사항

Trident를 배포하기 전에 호스트에 필요한 프로토콜 사전 요구 사항을 설치하고 구성해야 합니다.

요구 사항을 확인하십시오

- 배포가 모든 "요구사항"를 충족하는지 확인하십시오.
- 지원되는 버전의 Docker가 설치되어 있는지 확인하십시오. Docker 버전이 최신 버전이 아닌 경우 "설치하거나 업데이트하십시오".

```
docker --version
```

- 프로토콜 사전 요구 사항이 호스트에 설치 및 구성되었는지 확인하십시오.

NFS 도구

사용 중인 운영 체제에 맞는 명령을 사용하여 NFS 툴을 설치합니다.

RHEL 8+

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

Ubuntu

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

NFS 도구를 설치한 후 워커 노드를 재부팅하여 컨테이너에 볼륨을 연결할 때 발생하는 오류를 방지하십시오.

iSCSI 도구

운영 체제에 맞는 명령을 사용하여 iSCSI 툴을 설치합니다.

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

```
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 확인하십시오.

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

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 활성화:

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

> 반드시 `etc/multipath.conf`에 `find_multipaths no`가 `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `iscsid` 및 `multipathd`가 실행 중인지 확인하십시오.

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

6. 활성화 및 시작 `iscsi`:

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

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

```
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 확인하십시오.

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

3. 스캔을 수동으로 설정:

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

4. 다중 경로 지정 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
```

> 반드시 `etc/multipath.conf`에 `find_multipaths no`가 `defaults` 아래에 포함되어 있는지 확인하십시오.

5. `open-iscsi` 및 `multipath-tools`가 활성화되어 실행 중인지 확인하십시오.

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

NVMe 도구

사용 중인 운영 체제에 맞는 명령을 사용하여 NVMe 툴을 설치하십시오.

- NVMe에는 RHEL 9 이상이 필요합니다.
- Kubernetes 노드의 커널 버전이 너무 오래되었거나 해당 커널 버전에 맞는 NVMe 패키지를 사용할 수 없는 경우 노드의 커널 버전을 NVMe 패키지가 포함된 버전으로 업데이트해야 할 수 있습니다.

RHEL 9

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

Ubuntu

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

FC 툴

운영 체제에 대한 명령을 사용하여 FC 툴을 설치합니다.

- FC PV를 사용하는 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 워커 노드를 사용할 때, 인라인 공간 재확보를 수행하려면 `discard` StorageClass에서 mountOption을 지정하십시오. "Red Hat 문서"를 참조하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치하십시오.

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

2. 다중 경로 지정 활성화:

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

> 반드시 `etc/multipath.conf`에 `find_multipaths no`가 `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipathd`이(가) 실행 중인지 확인하십시오.

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

Ubuntu

1. 다음 시스템 패키지를 설치하십시오.

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

2. 다중 경로 지정 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
```

> 반드시 `etc/multipath.conf`에 `find_multipaths no`가 `defaults` 아래에 포함되어 있는지 확인하십시오.

3. `multipath-tools`이(가) 활성화되어 실행 중인지 확인하십시오.

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

Trident 구축

Trident for Docker는 NetApp 스토리지 플랫폼을 위한 Docker 에코시스템과의 직접적인 통합을 제공합니다. 스토리지 플랫폼에서 Docker 호스트로 스토리지 리소스를 프로비저닝하고 관리할 수 있도록 지원하며, 향후 다른 플랫폼을 추가할 수 있는 프레임워크도 제공합니다.

Trident의 여러 인스턴스를 동일한 호스트에서 동시에 실행할 수 있습니다. 이를 통해 여러 스토리지 시스템 및 스토리지 유형에 동시에 연결할 수 있으며 Docker 볼륨에 사용되는 스토리지를 사용자 지정할 수 있습니다.

필요한 것

"배포를 위한 사전 요구 사항"를 참조하십시오. 사전 요구 사항이 충족되었는지 확인한 후 Trident를 배포할 준비가 된 것입니다.

Docker 관리형 플러그인 방식(버전 1.13/17.03 이상)

시작하기 전에

기존 데몬 방식으로 Docker 1.13/17.03 이전 버전의 Trident를 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 재시작합니다.

```
systemctl restart docker
```

3. Docker Engine 17.03(새 1.13) 이상이 설치되어 있는지 확인하십시오.

```
docker --version
```

사용 중인 버전이 오래된 경우 "설치 또는 설치 업데이트".

단계

1. 구성 파일을 생성하고 다음과 같이 옵션을 지정하십시오.
   - `config`: 기본 파일 이름은 `config.json`이지만 `config` 옵션에 파일 이름을 지정하여 원하는 이름을 사용할 수 있습니다. 구성 파일은 호스트 시스템의 `/etc/netappdvp` 디렉터리에 있어야 합니다.
   - `log-level`: 로깅 수준을 지정합니다((`debug`, `info`, `warn`, `error`, `fatal`). 기본값은 `info`입니다.
   - `debug`: 디버그 로깅 활성화 여부를 지정합니다. 기본값은 false입니다. true로 설정하면 log-level을 재정의합니다.
     i. 구성 파일의 위치를 생성합니다.

```
sudo mkdir -p /etc/netappdvp
```

ii. 구성 파일을 생성합니다.

```
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. 관리형 플러그인 시스템을 사용하여 Trident를 시작합니다. `<version>`를 사용 중인 플러그인 버전(xxx.xx.x)으로 교체합니다.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. 구성된 시스템에서 스토리지를 사용하려면 Trident를 사용하기 시작하십시오.

a. "firstVolume"이라는 이름의 볼륨을 생성합니다.

```
docker volume create -d netapp --name firstVolume
```

b. 컨테이너 시작 시 기본 볼륨 생성:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. "firstVolume" 볼륨을 제거하세요:

```
docker volume rm firstVolume
```

기존 방식(버전 **1.12** 이하)

시작하기 전에

1. Docker 버전 1.10 이상이 있는지 확인하십시오.

```
docker --version
```

사용 중인 버전이 오래된 경우 설치를 업데이트하십시오.

```
curl -fsSL https://get.docker.com/ | sh
```

또는 "배포 지침을 따르십시오".

2. 시스템에 NFS 및/또는 iSCSI가 구성되어 있는지 확인하십시오.

단계

1. NetApp Docker 볼륨 플러그인을 설치하고 구성하십시오.
   a. 애플리케이션을 다운로드하고 압축을 풉니다.

```
wget
https://github.com/NetApp/trident/releases/download/10.0/trident-
installer-25.10.0.tar.gz
tar zxf trident-installer-25.10.0.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 생성합니다.

```
sudo mkdir -p /etc/netappdvp
```

   d. 구성 파일을 생성합니다.

```
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 생성한 후 원하는 구성 파일을 사용하여 Trident 데몬을 시작합니다.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

별도로 지정하지 않으면 볼륨 드라이버의 기본 이름은 "netapp"입니다.

데몬이 시작되면 Docker CLI 인터페이스를 사용하여 볼륨을 생성하고 관리할 수 있습니다.

3. 볼륨 생성:

```
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 Trident를 시작합니다

systemd 기반 시스템용 샘플 유닛 파일은 Git 저장소의 `contrib/trident.service.example`에서 찾을 수 있습니다. RHEL에서 파일을 사용하려면 다음을 수행합니다.

1. 파일을 올바른 위치에 복사합니다.

   여러 인스턴스를 실행하는 경우 유닛 파일에 고유한 이름을 사용해야 합니다.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. 파일을 편집하고 설명(2행)을 드라이버 이름과 일치하도록 변경하고 구성 파일 경로(9행)를 환경에 맞게 변경하십시오.
3. 변경 사항을 수집하도록 systemd를 다시 로드합니다.

   ```
   systemctl daemon-reload
   ```

4. 서비스를 활성화합니다.

   이 이름은 `/usr/lib/systemd/system` 디렉터리에서 파일 이름을 지정한 내용에 따라 달라집니다.

   ```
   systemctl enable trident
   ```

5. 서비스를 시작합니다.

   ```
   systemctl start trident
   ```

6. 상태를 확인합니다.

   ```
   systemctl status trident
   ```

ⓘ 유닛 파일을 수정할 때마다 변경 사항을 인식하도록 `systemctl daemon-reload` 명령을 실행합니다.

Trident 업그레이드 또는 제거

Trident for Docker를 안전하게 업그레이드할 수 있으며, 사용 중인 볼륨에는 아무런 영향을 미치지 않습니다. 업그레이드 과정 중에는 `docker volume` 플러그인에 대한 명령이 잠시 실패하고, 플러그인이 다시 실행될 때까지 애플리케이션에서 볼륨을 마운트할 수 없는 시간이 발생할 수 있습니다. 대부분의 경우 이 시간은 몇 초에 불과합니다.

업그레이드

Docker용 Trident를 업그레이드하려면 아래 단계를 수행하십시오.

단계

1. 기존 볼륨을 나열합니다.

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. 플러그인을 비활성화합니다.

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 플러그인 업그레이드:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> Trident의 18.01 릴리스는 nDVP를 대체합니다. `netapp/ndvp-plugin` 이미지에서 `netapp/trident-plugin` 이미지로 직접 업그레이드해야 합니다.

4. 플러그인을 활성화합니다.

```
docker plugin enable netapp:latest
```

5. 플러그인이 활성화되어 있는지 확인합니다.

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. 볼륨이 표시되는지 확인합니다.

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

> 이전 버전의 Trident(20.10 이전)에서 Trident 20.10 이상으로 업그레이드하는 경우 오류가 발생할 수 있습니다. 자세한 내용은 "알려진 문제"를 참조하십시오. 오류가 발생하는 경우 먼저 플러그인을 비활성화한 다음 플러그인을 제거하고 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 Trident를 제거하려면 아래 단계를 수행하십시오.

단계

1. 플러그인이 생성한 볼륨을 모두 제거하십시오.
2. 플러그인을 비활성화합니다.

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 플러그인을 제거합니다.

```
docker plugin rm netapp:latest
```

볼륨 작업

필요에 따라 Trident 드라이버 이름을 지정하여 표준 `docker volume` 명령을 사용하면 볼륨을 쉽게 생성, 복제 및 제거할 수 있습니다.

볼륨 생성

- 기본 이름을 사용하여 드라이버로 볼륨 생성:

```
docker volume create -d netapp --name firstVolume
```

- 특정 Trident 인스턴스를 사용하여 볼륨 생성:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

"옵션"을 지정하지 않으면 드라이버의 기본값이 사용됩니다.

- 기본 볼륨 크기를 재정의합니다. 드라이버를 사용하여 20GiB 볼륨을 생성하는 방법은 다음 예제를 참조하십시오.

```
docker volume create -d netapp --name my_vol --opt size=20G
```

볼륨 크기는 정수 값과 선택적으로 단위를 포함하는 문자열로 표현됩니다(예: 10G, 20GB, 3TiB). 단위가 지정되지 않은 경우 기본값은 G입니다. 크기 단위는 2의 거듭제곱(B, KiB, MiB, GiB, TiB) 또는 10의 거듭제곱(B, KB, MB, GB, TB)으로 표현할 수 있습니다. 간략한 단위 표기에는 2의 거듭제곱이 사용됩니다(G = GiB, T = TiB 등).

볼륨 제거

- 다른 Docker 볼륨과 마찬가지로 볼륨을 제거하십시오:

```
docker volume rm firstVolume
```

`solidfire-san` 드라이버를 사용할 때 위의 예에서는 볼륨을 삭제하고 제거합니다.

Docker용 Trident를 업그레이드하려면 아래 단계를 수행하십시오.

볼륨 복제

`ontap-nas`, `ontap-san` 및 `solidfire-san` 스토리지 드라이버를 사용하는 경우 Trident는 볼륨을 복제할 수 있습니다. `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버를 사용하는 경우에는 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새 스냅샷이 생성됩니다.

- 볼륨을 검사하여 스냅샷을 열거합니다.

```
docker volume inspect <volume_name>
```

- 기존 볼륨에서 새 볼륨을 생성합니다. 이렇게 하면 새 스냅샷이 생성됩니다.

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- 볼륨의 기존 스냅샷에서 새 볼륨을 생성합니다. 새 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

컨테이너는 Trident를 사용하여 외부에서 생성된 블록 디바이스(또는 해당 클론)에 액세스할 수 있습니다. 단, 해당 디바이스에 파티션이 없고 파일 시스템이 Trident에서 지원되는 경우에만 가능합니다(예: `ext4`로 포맷된 `/dev/sdc1` 디바이스는 Trident를 통해 액세스할 수 없음).

드라이버별 볼륨 옵션

각 스토리지 드라이버는 볼륨 생성 시 지정할 수 있는 다양한 옵션 세트를 제공하여 결과물을 사용자 지정할 수 있습니다. 구성된 스토리지 시스템에 적용되는 옵션은 아래를 참조하십시오.

볼륨 생성 작업 중에 이러한 옵션을 사용하는 것은 간단합니다. CLI 작업 중에 `-o` 연산자를 사용하여 옵션과 값을 제공합니다. 이러한 값은 JSON 구성 파일의 동등한 값을 재정의합니다.

ONTAP 볼륨 옵션

NFS, iSCSI 및 FC에 대한 볼륨 생성 옵션은 다음과 같습니다.

옵션	설명
`size`	볼륨의 크기로, 기본값은 1GiB입니다.
`spaceReserve`	볼륨을 씬 프로비저닝 또는 씩 프로비저닝할지 결정하며, 기본값은 씬 프로비저닝입니다. 유효한 값은 `none`(씬 프로비저닝)과 `volume`(씩 프로비저닝)입니다.
`snapshotPolicy`	이렇게 하면 스냅샷 정책이 원하는 값으로 설정됩니다. 기본값은 `none`이며, 볼륨에 대해 스냅샷이 자동으로 생성되지 않음을 의미합니다. 스토리지 관리자가 수정하지 않는 한, 모든 `ONTAP` 시스템에는 `"default"`라는 정책이 존재하며, 이 정책은 `6`개의 시간별, `2`개의 일별, `2`개의 주별 스냅샷을 생성하고 보존합니다. 스냅샷에 보존된 데이터는 볼륨 내 모든 디렉터리의 `.snapshot` 디렉터리를 탐색하여 복구할 수 있습니다.
`snapshotReserve`	이렇게 하면 스냅샷 예약 공간이 원하는 비율로 설정됩니다. 기본값은 값이 없음을 의미하며, snapshotPolicy를 선택한 경우 ONTAP가 snapshotReserve(일반적으로 5%)를 선택하고, snapshotPolicy가 none인 경우 0%를 선택합니다. 모든 ONTAP 백엔드에 대해 구성 파일에서 기본 snapshotReserve 값을 설정할 수 있으며, ontap-nas-economy를 제외한 모든 ONTAP 백엔드의 볼륨 생성 옵션으로 사용할 수 있습니다.

옵션	설명
splitOnClone	볼륨을 복제할 때 이 옵션을 선택하면 ONTAP가 복제본을 상위 볼륨에서 즉시 분할합니다. 기본값은 `false`입니다. 볼륨 복제를 위한 일부 사용 사례는 스토리지 효율성을 위한 기회가 거의 없기 때문에 생성 즉시 복제본을 상위 볼륨에서 분할하는 것이 가장 좋습니다. 예를 들어, 빈 데이터베이스를 복제하면 시간은 크게 절약할 수 있지만 스토리지 절약 효과는 거의 없으므로 복제본을 즉시 분할하는 것이 가장 좋습니다.
encryption	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨에 NAE가 활성화됩니다. 자세한 내용은 "Trident가 NVE 및 NAE와 작동하는 방식"을(를) 참조하십시오.
tieringPolicy	볼륨에 사용할 티어링 정책을 설정합니다. 이 정책은 데이터가 비활성화(콜드)될 때 클라우드 티어로 이동할지 여부를 결정합니다.

다음 추가 옵션은 NFS *전용*입니다.

옵션	설명
unixPermissions	이 설정은 볼륨 자체에 대한 권한 설정을 제어합니다. 기본적으로 권한은 `---rwxr-xr-x 또는 숫자 표기법으로 0755로 설정되며, `root`가 소유자입니다. 텍스트 또는 숫자 형식 모두 사용 가능합니다.
snapshotDir	이 값을 `true`로 설정하면 `.snapshot 디렉터리가 볼륨에 액세스하는 클라이언트에 표시됩니다. 기본값은 `false`이며, 이는 `.snapshot 디렉터리의 가시성이 기본적으로 비활성화되어 있음을 의미합니다. 일부 이미지(예: 공식 MySQL 이미지)는 .snapshot 디렉터리가 표시될 때 예상대로 작동하지 않습니다.
exportPolicy	볼륨에 사용할 내보내기 정책을 설정합니다. 기본값은 `default`입니다.
securityStyle	볼륨 접근에 사용할 보안 스타일을 설정합니다. 기본값은 `unix`입니다. 유효한 값은 `unix 및 `mixed`입니다.

다음 추가 옵션은 iSCSI *전용*입니다.

옵션	설명
`fileSystemType`	iSCSI 볼륨 포맷에 사용되는 파일 시스템을 설정합니다. 기본값은 `ext4`입니다. 유효한 값은 `ext3`, `ext4` 및 `xfs`입니다.
`spaceAllocation`	이 값을 `false`로 설정하면 LUN의 공간 할당 기능이 비활성화됩니다. 기본값은 `true`이며, ONTAP이 볼륨의 공간이 부족해져 해당 볼륨의 LUN에서 쓰기 작업을 더 이상 수행할 수 없을 때 호스트에 알림을 보낸다는 의미입니다. 또한 이 옵션을 사용하면 호스트에서 데이터를 삭제할 때 ONTAP이 자동으로 공간을 회수할 수 있습니다.

예

아래 예를 참조하십시오.

- 10 GiB 볼륨을 생성합니다.

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 스냅샷이 있는 100GiB 볼륨 생성:

```
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 예약량이 지정되지 않고 스냅샷 정책이 `none`인 경우 Trident는 0%의 스냅샷 예약량을 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 예약률이 10%인 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- 스냅샷 정책과 10%의 사용자 지정 스냅샷 예약 공간을 사용하여 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- 스냅샷 정책이 적용된 볼륨을 생성하고 ONTAP의 기본 스냅샷 예약값(일반적으로 5%)을 수락하십시오.

```
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 표시합니다. 볼륨을 생성할 때 해당 볼륨과 연결된 QoS 정책은 `-o type=service_level` 명명법을 사용하여 지정됩니다.

Element 드라이버를 사용하여 QoS 서비스 수준을 정의하는 첫 번째 단계는 하나 이상의 유형을 생성하고 구성 파일에서 이름과 연결된 최소, 최대 및 버스트 IOPS를 지정하는 것입니다.

기타 Element 소프트웨어 볼륨 생성 옵션은 다음과 같습니다.

옵션	설명
`size`	볼륨의 크기는 기본값으로 1GiB 또는 구성 항목 … "기본값": {"size": "5G"}.
`blocksize`	512 또는 4096 중 하나를 사용하십시오. 기본값은 512 또는 구성 항목 DefaultBlockSize입니다.

예

다음은 QoS 정의가 포함된 샘플 구성 파일입니다.

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

위 구성에는 Bronze, Silver, Gold라는 세 가지 정책 정의가 있습니다. 이 이름은 임의로 지정한 것입니다.

- 10GiB Gold 볼륨을 생성합니다.

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 100 GiB Bronze 볼륨 생성:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

로그 수집

문제 해결에 도움이 되도록 로그를 수집할 수 있습니다. 로그 수집 방법은 Docker 플러그인을 실행하는 방식에 따라 다릅니다.

문제 해결을 위해 로그를 수집합니다

단계

1. 권장되는 관리형 플러그인 방식(즉, `docker plugin` 명령을 사용)을 사용하여 Trident를 실행하는 경우 다음과 같이 확인합니다.

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

   기본 로깅 수준을 사용하면 대부분의 문제를 진단할 수 있습니다. 만약 그것으로 충분하지 않다고 판단되면 디버그 로깅을 활성화할 수 있습니다.

2. 디버그 로깅을 활성화하려면 디버그 로깅이 활성화된 플러그인을 설치하십시오.

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   또는 플러그인이 이미 설치된 경우 디버그 로깅을 활성화하십시오.

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. 호스트에서 바이너리 자체를 실행하는 경우 로그는 호스트의 `/var/log/netappdvp` 디렉터리에서 확인할 수 있습니다. 디버그 로깅을 활성화하려면 플러그인을 실행할 때 `-debug`를 지정하십시오.

일반적인 문제 해결 팁

- 새로운 사용자들이 가장 흔히 겪는 문제는 플러그인 초기화를 방해하는 잘못된 구성입니다. 이러한 경우 플러그인을 설치하거나 활성화하려고 할 때 다음과 같은 메시지가 표시될 수 있습니다:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  이는 플러그인이 시작되지 않았음을 의미합니다. 다행히 이 플러그인은 포괄적인 로깅 기능을 갖추고 있어 발생할 수 있는 대부분의 문제를 진단하는 데 도움이 될 것입니다.

- 컨테이너에 PV를 마운트하는 데 문제가 있는 경우 `rpcbind`가 설치되어 실행 중인지 확인하십시오. 호스트 OS에 필요한 패키지 관리자를 사용하여 `rpcbind`가 실행 중인지 확인하십시오. `rpcbind` 서비스의 상태는 `systemctl status rpcbind` 또는 이와 동등한 명령을 실행하여 확인할 수 있습니다.

여러 Trident 인스턴스를 관리합니다

여러 스토리지 구성을 동시에 사용하려면 Trident의 여러 인스턴스가 필요합니다. 여러 인스턴스의 핵심은 컨테이너화된 플러그인의 `--alias` 옵션을 사용하거나 호스트에서 Trident를 인스턴스화할 때 `--volume-driver` 옵션을 사용하여 각 인스턴스에 서로 다른 이름을 지정하는 것입니다.

Docker 관리형 플러그인(버전 1.13/17.03 이상) 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. 별칭을 드라이버 이름으로 지정하여 볼륨을 생성합니다.

   예를 들어 금 볼륨의 경우:

   ```
   docker volume create -d gold --name ntapGold
   ```

   예를 들어, silver 볼륨의 경우:

   ```
   docker volume create -d silver --name ntapSilver
   ```

기존 버전(1.12 이하)의 단계

1. 사용자 지정 드라이버 ID를 사용하는 NFS 구성으로 플러그인을 실행하세요.

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 사용자 지정 드라이버 ID를 사용하는 iSCSI 구성으로 플러그인을 실행하십시오.

```
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.

전역 구성 옵션

이러한 구성 옵션은 사용 중인 스토리지 플랫폼과 관계없이 모든 Trident 구성에 적용됩니다.

옵션	설명	예
`version`	구성 파일 버전 번호	`1`
`storageDriverName`	스토리지 드라이버 이름	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	볼륨 이름에 사용할 수 있는 선택적 접두사입니다. 기본값: `netappdvp_`.	`staging_`

옵션	설명	예
`limitVolumeSize`	볼륨 크기에 대한 선택적 제한입니다. 기본값: "" (적용되지 않음)	`10g`

Element 백엔드에는 `storagePrefix`(기본값 포함)을 사용하지 마십시오. 기본적으로 `solidfire-san` 드라이버는 이 설정을 무시하고 접두사를 사용하지 않습니다. NetApp에서는 Docker 볼륨 매핑에 특정 tenantID를 사용하거나 이름 변환이 사용된 경우 Docker 버전, 드라이버 정보 및 Docker의 원시 이름으로 채워진 속성 데이터를 사용할 것을 권장합니다.

볼륨을 생성할 때마다 설정을 일일이 지정할 필요 없이 기본 옵션을 사용할 수 있습니다. `size` 옵션은 모든 컨트롤러 유형에서 사용할 수 있습니다. 기본 볼륨 크기를 설정하는 방법에 대한 예시는 ONTAP 구성 섹션을 참조하십시오.

옵션	설명	예
`size`	새 볼륨의 선택적 기본 크기입니다. 기본값: `1G`	`10G`

ONTAP 구성

위의 전역 구성 값 외에도 ONTAP를 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 주소입니다. **ONTAP NAS** 드라이버: NetApp에서는 `dataLIF`를 지정하는 것을 권장합니다. 지정하지 않으면 Trident가 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 생성할 수 있습니다. **ONTAP SAN** 드라이버: iSCSI 또는 FC에 대해 지정하지 마십시오. Trident는 "ONTAP 선택적 LUN 매핑"를 사용하여 다중 경로 세션을 설정하는 데 필요한 iSCSI 또는 FC LIF를 검색합니다. `dataLIF`가 명시적으로 정의된 경우 경고가 생성됩니다.	`10.0.0.2`

옵션	설명	예
svm	사용할 스토리지 가상 머신(관리 LIF가 클러스터 LIF인 경우 필수)	svm_nfs
username	스토리지 장치에 연결할 사용자 이름	vsadmin
password	스토리지 장치에 연결하기 위한 암호	secret
aggregate	프로비저닝용 애그리게이트(선택 사항, 설정된 경우 SVM에 할당되어야 함). ontap-nas-flexgroup 드라이버의 경우 이 옵션은 무시됩니다. SVM에 할당된 모든 애그리게이트는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다.	aggr1
limitAggregateUsage	선택 사항이며, 사용량이 이 비율을 초과하면 프로비저닝이 실패합니다	75%
nfsMountOptions	NFS 마운트 옵션을 세밀하게 제어할 수 있습니다. 기본값은 "-o nfsvers=3"입니다. * ontap-nas 및 ontap-nas-economy 드라이버에서만 사용 가능합니다*. "NFS 호스트 구성 정보는 여기를 참조하십시오"	-o nfsvers=4
igroupName	Trident는 노드별 igroups `netappdvp`로 생성 및 관리합니다. 이 값은 변경하거나 생략할 수 없습니다. • ontap-san 드라이버에만 사용할 수 있습니다*.	netappdvp
limitVolumeSize	최대 요청 가능 볼륨 크기.	300g
qtreesPerFlexvol	FlexVol당 최대 qtree 수는 [50, 300] 범위 내에 있어야 하며 기본값은 200입니다. • ontap-nas-economy 드라이버의 경우, 이 옵션을 사용하면 FlexVol당 최대 qtree 수를 사용자 지정할 수 있습니다*.	300

옵션	설명	예
sanType	* ontap-san 드라이버에서만 지원됩니다.* `iscsi`를 사용하여 iSCSI, `nvme`를 사용하여 NVMe/TCP 또는 `fcp`를 사용하여 Fibre Channel(FC)을 통한 SCSI를 선택합니다.	iscsi 공백인 경우
limitVolumePoolSize	* ontap-san-economy 및 ontap-san-economy 드라이버에서만 지원됩니다.* ONTAP ontap-nas-economy 및 ontap-san-economy 드라이버에서 FlexVol 크기를 제한합니다.	300g

볼륨을 생성할 때마다 지정할 필요가 없도록 기본 옵션을 사용할 수 있습니다.

옵션	설명	예
spaceReserve	공간 예약 모드; none(씬 프로비저닝) 또는 volume(씩 프로비저닝)	none
snapshotPolicy	사용할 스냅샷 정책, 기본값은 none	none
snapshotReserve	스냅샷 예약 비율, 기본값은 ""이며 ONTAP 기본값을 수락합니다	10
splitOnClone	생성 시 상위 클론에서 클론을 분할합니다. 기본값은 false	false
encryption	새 볼륨에서 NetApp Volume Encryption(NVE)을 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이센스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨에 NAE가 활성화됩니다. 자세한 내용은 "Trident가 NVE 및 NAE와 작동하는 방식"을(를) 참조하십시오.	true
unixPermissions	프로비저닝된 NFS 볼륨에 대한 NAS 옵션, 기본값 777	777
snapshotDir	`.snapshot` 디렉토리에 액세스하기 위한 NAS 옵션.	NFSv4의 경우 "true", NFSv3의 경우 "false"

옵션	설명	예
exportPolicy	사용할 NFS 엑스포트 정책에 대한 NAS 옵션, 기본값은 default	default
securityStyle	프로비저닝된 NFS 볼륨에 액세스하기 위한 NAS 옵션입니다. NFS는 mixed 및 unix 보안 스타일을 지원합니다. 기본값은 `unix`입니다.	unix
fileSystemType	파일 시스템 유형을 선택하는 SAN 옵션, 기본값은 ext4	xfs
tieringPolicy	사용할 계층화 정책, 기본값은 `none`입니다.	none
skipRecoveryQueue	볼륨 삭제 시 스토리지의 복구 큐를 건너뛰고 볼륨을 즉시 삭제합니다.	``

확장 옵션

`ontap-nas` 및 `ontap-san` 드라이버는 각 Docker 볼륨에 대해 ONTAP FlexVol을 생성합니다. ONTAP는 클러스터 노드당 최대 1,000개의 FlexVols를 지원하며 클러스터 최대 12,000개의 FlexVol 볼륨을 지원합니다. Docker 볼륨 요구 사항이 이 제한 내에 있는 경우 `ontap-nas` 드라이버는 Docker 볼륨 단위 스냅샷 및 클론 생성과 같은 FlexVols에서 제공하는 추가 기능으로 인해 선호되는 NAS 솔루션입니다.

FlexVol 제한으로 수용할 수 있는 것보다 더 많은 Docker 볼륨이 필요한 경우 ontap-nas-economy 또는 ontap-san-economy 드라이버를 선택하십시오.

`ontap-nas-economy` 드라이버는 자동으로 관리되는 FlexVol 볼륨 풀 내에서 ONTAP Qtree로 Docker 볼륨을 생성합니다. Qtree는 클러스터 노드당 최대 100,000개, 클러스터당 최대 2,400,000개까지 훨씬 더 큰 확장성을 제공하지만 일부 기능이 제한됩니다. `ontap-nas-economy` 드라이버는 Docker 볼륨 단위의 스냅샷 또는 클론 생성을 지원하지 않습니다.

ontap-nas-economy 드라이버는 Docker Swarm이 여러 노드에 걸쳐 볼륨 생성을 조정하지 않기 때문에 현재 Docker Swarm에서 지원되지 않습니다.

`ontap-san-economy` 드라이버는 자동으로 관리되는 FlexVol 볼륨의 공유 풀 내에서 Docker 볼륨을 ONTAP LUN으로 생성합니다. 이러한 방식으로 각 FlexVol은 하나의 LUN으로만 제한되지 않으며 SAN 워크로드에 더 나은 확장성을 제공합니다. 스토리지 어레이에 따라 ONTAP는 클러스터당 최대 16384개의 LUN을 지원합니다. 볼륨이 내부적으로 LUN이기 때문에 이 드라이버는 Docker 볼륨 단위의 스냅샷 및 클론 생성을 지원합니다.

`ontap-nas-flexgroup` 드라이버를 선택하여 수십억 개의 파일로 페타바이트 범위까지 확장할 수 있는 단일 볼륨에 대한 병렬 처리를 향상시키십시오. FlexGroups의 이상적인 사용 사례로는 AI/ML/DL, 빅 데이터 및 분석, 소프트웨어 빌드, 스트리밍, 파일 리포지토리 등이 있습니다. Trident는 FlexGroup 볼륨을 프로비저닝할 때 SVM에 할당된 모든 애그리게이트를 사용합니다. Trident의 FlexGroup 지원에는 다음과 같은 고려 사항도 있습니다.

- ONTAP 버전 9.2 이상이 필요합니다.
- 현재(작성 시점 기준) FlexGroups는 NFS v3만 지원합니다.
- SVM에 대해 64비트 NFSv3 식별자를 활성화하는 것이 좋습니다.
- 최소 권장 FlexGroup 멤버/볼륨 크기는 100GiB입니다.
- FlexGroup 볼륨은 복제가 지원되지 않습니다.

FlexGroups 및 FlexGroups에 적합한 워크로드에 대한 자세한 내용은 "NetApp FlexGroup 볼륨 모범 사례 및 구현 가이드"을 참조하십시오.

고급 기능과 대규모 확장성을 동일한 환경에서 구현하려면 Docker Volume Plugin의 여러 인스턴스를 실행할 수 있습니다. 하나는 `ontap-nas`을 사용하고 다른 하나는 `ontap-nas-economy`을 사용합니다.

Trident용 사용자 지정 **ONTAP** 역할

Trident에서 작업을 수행하기 위해 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 권한으로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident는 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 "Trident 사용자 지정 역할 생성기"을 참조하십시오.

ONTAP CLI 사용

1. 다음 명령을 사용하여 새 역할을 생성합니다.

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

2. Trident 사용자의 사용자 이름을 생성합니다.

```
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 역할 매핑:

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

System Manager 사용

ONTAP System Manager에서 다음 단계를 수행하십시오.

1. 사용자 지정 역할 생성:
   a. 클러스터 수준에서 사용자 지정 역할을 생성하려면 * Cluster > Settings * 를 선택합니다.

      (또는) SVM 수준에서 사용자 지정 역할을 생성하려면 *스토리지 > 스토리지 VM > `required SVM`> 설정 > 사용자 및 역할*을 선택하십시오.

   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 아래에서 추가 아이콘 *+*를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 역할 드롭다운 메뉴에서 역할을 선택합니다.
   c. *저장*을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

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

ONTAP 구성 파일의 예

<code>ontap-nas</code> 드라이버의 NFS 예

```
{
  "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> 드라이버의 NFS 예

```
{
  "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> 드라이버의 **NFS** 예

```
{
  "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> 드라이버의 **iSCSI** 예

```
{
  "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> 드라이버의 **NFS** 예

```
{
  "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> 드라이버의 NVMe/TCP 예

```
{
  "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> 드라이버의 FC 기반 SCSI 예

```
{
  "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
}
```

Element 소프트웨어 구성

전역 구성 값 외에도 Element 소프트웨어(NetApp HCI/SolidFire)를 사용할 때 다음과 같은 옵션을 사용할 수 있습니다.

옵션	설명	예
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	iSCSI IP 주소 및 포트	10.0.0.7:3260
`TenantName`	사용할 SolidFire 테넌트(찾을 수 없으면 생성됨)	`docker`

옵션	설명	예
`InitiatorIFace`	iSCSI 트래픽을 기본이 아닌 인터페이스로 제한할 때 인터페이스를 지정합니다	`default`
`Types`	QoS 사양	아래 예를 참조하십시오
`LegacyNamePrefix`	업그레이드된 Trident 설치에 사용되는 접두사입니다. 1.3.2 이전 버전의 Trident를 사용했고 기존 볼륨으로 업그레이드를 수행하는 경우 volume-name 방법을 통해 매핑된 이전 볼륨에 액세스하려면 이 값을 설정해야 합니다.	`netappdvp-`

`solidfire-san` 드라이버는 `Docker Swarm`을 지원하지 않습니다.

Element 소프트웨어 구성 파일의 예

```
{
  "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 Trident를 사용할 때 발생하는 알려진 문제 및 제한 사항에 대한 정보를 찾아보세요.

Trident Docker Volume Plugin을 이전 버전에서 **20.10** 이상 버전으로 업그레이드하면 **no such file or directory** 오류와 함께 업그레이드가 실패합니다.

해결 방법

1. 플러그인을 비활성화합니다.

```
docker plugin disable -f netapp:latest
```

2. 플러그인을 제거합니다.

```
docker plugin rm -f netapp:latest
```

3. 추가 `config` 매개변수를 제공하여 플러그인을 다시 설치하십시오.

```
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 한 글자로 된 이름을 Windows 경로로 해석합니다 "버그 25773을 참조하십시오".

Docker Swarm에는 Trident가 모든 스토리지 및 드라이버 조합에서 이를 지원하지 못하도록 하는 특정 동작이 있습니다.

- Docker Swarm은 현재 고유 볼륨 식별자로 볼륨 ID 대신 볼륨 이름을 사용합니다.
- 볼륨 요청은 Swarm 클러스터의 각 노드로 동시에 전송됩니다.
- 볼륨 플러그인(Trident 포함)은 Swarm 클러스터의 각 노드에서 독립적으로 실행되어야 합니다. ONTAP 작동 방식과 `ontap-nas` 및 `ontap-san` 드라이버의 기능 특성상, 이러한 제약 조건 내에서 작동할 수 있는 플러그인은 해당 드라이버뿐입니다.

나머지 드라이버는 경쟁 조건과 같은 문제에 직면할 수 있으며, 이로 인해 명확한 "승자" 없이 단일 요청에 대해 다수의 볼륨이 생성될 수 있습니다. 예를 들어 Element에는 볼륨이 동일한 이름을 갖지만 서로 다른 ID를 가질 수 있도록 하는 기능이 있습니다.

NetApp은 Docker 팀에 피드백을 제공했지만 향후 조치에 대한 표시는 없습니다.

FlexGroup이 프로비저닝되는 동안 ONTAP는 두 번째 FlexGroup이 프로비저닝 중인 FlexGroup과 하나 이상의 애그리게이트를 공유하는 경우 두 번째 FlexGroup을 프로비저닝하지 않습니다.

모범 사례 및 권장 사항

구축

Trident를 배포할 때 여기에 나열된 권장 사항을 사용하십시오.

전용 네임스페이스에 배포

"네임스페이스"서로 다른 애플리케이션 간의 관리적 분리를 제공하고 리소스 공유를 방해하는 요소가 될 수 있습니다. 예를 들어, 한 네임스페이스의 PVC는 다른 네임스페이스에서 사용할 수 없습니다. Trident는 Kubernetes 클러스터의 모든 네임스페이스에 PV 리소스를 제공하며, 따라서 높은 권한을 가진 서비스 계정을 활용합니다.

또한 Trident Pod에 액세스하면 사용자가 스토리지 시스템 자격 증명 및 기타 민감한 정보에 액세스할 수 있습니다. 애플리케이션 사용자와 관리 애플리케이션이 Trident 객체 정의 또는 Pod 자체에 액세스할 수 없도록 하는 것이 중요합니다.

할당량 및 범위 제한을 사용하여 스토리지 사용량 제어

Kubernetes에는 애플리케이션의 리소스 사용량을 제한하는 강력한 메커니즘을 제공하는 두 가지 기능이 있습니다. "스토리지 할당량 메커니즘"를 사용하면 관리자가 네임스페이스별로 전역 및 스토리지 클래스별 용량 및 객체 개수 사용량 제한을 구현할 수 있습니다. 또한 "범위 제한"를 사용하면 PVC 요청이 프로비저너로 전달되기 전에 최소값과 최대값 범위 내에 있는지 확인할 수 있습니다.

이러한 값은 네임스페이스별로 정의되므로 각 네임스페이스는 리소스 요구 사항에 맞는 값을 정의해야 합니다. 에 대한 정보는 여기를 참조하십시오 "할당량을 활용하는 방법".

스토리지 구성

NetApp 포트폴리오의 각 스토리지 플랫폼은 컨테이너화된 애플리케이션이든 아니든 관계없이 애플리케이션에 도움이 되는 고유한 기능을 제공합니다.

플랫폼 개요

Trident는 ONTAP 및 Element와 함께 작동합니다. 모든 애플리케이션과 시나리오에 다른 것보다 더 적합한 플랫폼은 없지만, 플랫폼을 선택할 때는 애플리케이션의 요구 사항과 장치를 관리하는 팀을 고려해야 합니다.

사용하는 프로토콜과 호스트 운영 체제에 대한 기본 모범 사례를 따라야 합니다. 선택적으로, 특정 애플리케이션에 맞게 스토리지를 최적화하기 위해 백엔드, 스토리지 클래스 및 PVC 설정에 애플리케이션 모범 사례를 적용하는 것을 고려할 수 있습니다.

ONTAP 및 Cloud Volumes ONTAP 모범 사례

Trident용 ONTAP 및 Cloud Volumes ONTAP 구성 모범 사례를 알아보십시오.

다음 권장 사항은 Trident에서 동적으로 프로비저닝한 볼륨을 사용하는 컨테이너화된 워크로드에 대해 ONTAP를 구성하기 위한 지침입니다. 각 권장 사항은 사용 환경에 적합한지 검토하고 평가해야 합니다.

Trident 전용 SVM 사용

스토리지 가상 머신(SVM)은 ONTAP 시스템에서 테넌트 간의 격리 및 관리 분리를 제공합니다. 애플리케이션에 SVM을 전용으로 사용하면 권한 위임이 가능하고 리소스 사용량을 제한하기 위한 모범 사례를 적용할 수 있습니다.

SVM 관리에는 여러 가지 옵션이 있습니다.

- 백엔드 구성에서 클러스터 관리 인터페이스와 적절한 자격 증명을 제공하고 SVM 이름을 지정하십시오.
- ONTAP System Manager 또는 CLI를 사용하여 SVM용 전용 관리 인터페이스를 생성합니다.
- NFS 데이터 인터페이스와 관리 역할을 공유합니다.

각 경우에 인터페이스는 DNS에 등록되어 있어야 하며, Trident를 구성할 때 DNS 이름을 사용해야 합니다. 이렇게 하면 네트워크 ID 보존을 사용하지 않고도 SVM-DR과 같은 일부 재해 복구 시나리오를 구현할 수 있습니다.

SVM에 전용 관리 LIF를 사용할지 공유 관리 LIF를 사용할지 여부는 중요하지 않지만, 선택한 방식에 맞춰 네트워크 보안 정책을 수립해야 합니다. 관계없이, 관리 LIF는 DNS를 통해 액세스할 수 있어야 하며, "SVM-DR"Trident와 함께 사용될 경우 최대한의 유연성을 제공할 수 있습니다.

최대 볼륨 수 제한

ONTAP 스토리지 시스템에는 최대 볼륨 수가 있으며, 이는 소프트웨어 버전 및 하드웨어 플랫폼에 따라 다릅니다. 특정 플랫폼 및 ONTAP 버전에 대한 정확한 제한 사항을 확인하려면 "NetApp Hardware Universe"을 참조하십시오. 볼륨 수가 소진되면 Trident뿐만 아니라 모든 스토리지 요청에 대한 프로비저닝 작업이 실패합니다.

Trident의 `ontap-nas` 및 `ontap-san` 드라이버는 생성되는 각 Kubernetes 영구 볼륨(PV)에 대해 FlexVolume을 프로비저닝합니다. `ontap-nas-economy` 드라이버는 약 200개의 PV마다 하나의 FlexVolume을 생성합니다(50~300 사이에서 구성 가능). `ontap-san-economy` 드라이버는 약 100개의 PV마다 하나의 FlexVolume을 생성합니다(50~200 사이에서 구성 가능). Trident가 스토리지 시스템에서 사용 가능한 모든 볼륨을 소비하지 않도록 하려면 SVM에 제한을 설정해야 합니다. 명령줄에서 이 작업을 수행할 수 있습니다.

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

```
`max-volumes`의 값은 사용자 환경에 특정한 여러 기준에 따라 달라집니다.
```

- ONTAP 클러스터의 기존 볼륨 수
- 다른 애플리케이션에 대해 Trident 외부에서 프로비저닝할 것으로 예상되는 볼륨 수
- Kubernetes 애플리케이션에서 사용될 것으로 예상되는 영구 볼륨의 수

```
`max-volumes`값은 개별 ONTAP 노드가 아닌 ONTAP 클러스터의 모든 노드에 걸쳐 프로비저닝된 총 볼륨 수입니다. 따라서 ONTAP 클러스터 노드에 다른 노드보다 Trident 프로비저닝 볼륨이 훨씬 많거나 적을 수 있습니다.
```

예를 들어, 2노드 ONTAP 클러스터는 최대 2000개의 FlexVol 볼륨을 호스팅할 수 있습니다. 최대 볼륨 수를 1250으로 설정하는 것은 매우 적절해 보입니다. 그러나 한 노드에서만 "애그리게이트"이 SVM에 할당되거나, 한 노드에서 할당된 애그리게이트에 대해 프로비저닝이 불가능한 경우(예: 용량 부족), 다른 노드가 모든 Trident 프로비저닝 볼륨의 대상이

됩니다. 이는 `max-volumes` 값에 도달하기 전에 해당 노드의 볼륨 제한에 도달할 수 있음을 의미하며, Trident 및 해당 노드를 사용하는 다른 볼륨 작업 모두에 영향을 미칠 수 있습니다. 클러스터의 각 노드에서 **Trident**가 사용하는 **SVM**에 동일한 수의 애그리게이트가 할당되도록 하면 이러한 상황을 방지할 수 있습니다**.**

볼륨 복제

NetApp Trident는 `ontap-nas`, `ontap-san` 및 `solidfire-san` 스토리지 드라이버를 사용할 때 볼륨 복제를 지원합니다. `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버를 사용할 때는 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새 스냅샷이 생성됩니다.

> 다른 StorageClass와 연결된 PVC를 복제하지 마십시오. 호환성을 보장하고 예기치 않은 동작을 방지하려면 동일한 StorageClass 내에서 복제 작업을 수행하십시오.

Trident에서 생성한 볼륨의 최대 크기 제한

Trident에서 생성할 수 있는 볼륨의 최대 크기를 구성하려면, `limitVolumeSize` 매개변수를 `backend.json` 정의에 사용하십시오.

스토리지 어레이에서 볼륨 크기를 제어하는 것 외에도 Kubernetes 기능을 활용해야 합니다.

Trident에서 생성한 **FlexVols**의 최대 크기 제한

ontap-san-economy 및 ontap-nas-economy 드라이버의 풀로 사용되는 FlexVols의 최대 크기를 구성하려면, `limitVolumePoolSize` 매개변수를 `backend.json` 정의에 사용하십시오.

양방향 **CHAP**를 사용하도록 **Trident**를 구성합니다

백엔드 정의에서 CHAP 이니시에이터 및 대상 사용자 이름과 암호를 지정하고 Trident가 SVM에서 CHAP를 활성화하도록 설정할 수 있습니다. 백엔드 구성에서 `useCHAP` 매개 변수를 사용하면 Trident는 CHAP를 통해 ONTAP 백엔드에 대한 iSCSI 연결을 인증합니다.

SVM QoS 정책 생성 및 사용

SVM에 적용되는 ONTAP QoS 정책을 활용하면 Trident 프로비저닝된 볼륨이 소비할 수 있는 IOPS 수를 제한할 수 있습니다. 이는 "괴롭힘을 예방하다" 또는 제어 불능 상태의 컨테이너가 Trident SVM 외부의 워크로드에 영향을 미치는 것을 방지하는 데 도움이 됩니다.

몇 단계만 거치면 SVM에 대한 QoS 정책을 생성할 수 있습니다. 가장 정확한 정보는 사용 중인 ONTAP 버전의 설명서를 참조하십시오. 아래 예시는 SVM에서 사용 가능한 총 IOPS를 5000으로 제한하는 QoS 정책을 생성합니다.

```
# 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, 사용 중인 ONTAP 버전에서 지원하는 경우, 컨테이너화된 워크로드에 일정량의 처리량을 보장하기 위해 QoS

최소값을 사용하는 것을 고려할 수 있습니다. 적응형 QoS는 SVM 수준 정책과 호환되지 않습니다.

컨테이너화된 워크로드에 할당되는 IOPS 수는 여러 요소에 따라 달라집니다. 그중에서도 다음과 같은 요소들이 포함됩니다:

- 스토리지 어레이를 사용하는 다른 워크로드. Kubernetes 배포와 관련이 없는 다른 워크로드가 스토리지 리소스를 사용하는 경우, 해당 워크로드에 실수로 악영향을 미치지 않도록 주의해야 합니다.
- 컨테이너에서 실행될 것으로 예상되는 워크로드입니다. 높은 IOPS 요구 사항을 가진 워크로드가 컨테이너에서 실행될 경우, 낮은 QoS 정책은 사용자 경험 저하로 이어질 수 있습니다.

SVM 레벨에서 할당된 QoS 정책은 해당 SVM에 프로비저닝된 모든 볼륨이 동일한 IOPS 풀을 공유하게 된다는 점을 기억하는 것이 중요합니다. 컨테이너화된 애플리케이션 중 하나 또는 소수의 애플리케이션이 높은 IOPS 요구 사항을 가질 경우, 다른 컨테이너화된 워크로드에 과도한 부담을 줄 수 있습니다. 이러한 경우, 외부 자동화 도구를 사용하여 볼륨별 QoS 정책을 할당하는 것을 고려해 볼 수 있습니다.

ONTAP 버전이 9.8 이전인 경우에*만* QoS 정책 그룹을 SVM에 할당해야 합니다.

Trident용 **QoS** 정책 그룹 생성

QoS(서비스 품질)는 중요한 워크로드의 성능이 경쟁 워크로드로 인해 저하되지 않도록 보장합니다. ONTAP QoS 정책 그룹은 볼륨에 대한 QoS 옵션을 제공하며, 사용자가 하나 이상의 워크로드에 대한 처리량 상한을 정의할 수 있도록 합니다. QoS에 대한 자세한 내용은 "QoS를 통해 처리량 보장"을 참조하십시오. QoS 정책 그룹은 백엔드 또는 스토리지 풀에 지정할 수 있으며, 해당 풀 또는 백엔드에 생성된 각 볼륨에 적용됩니다.

ONTAP에는 기존 QoS 정책 그룹과 적응형 QoS 정책 그룹 두 가지 유형이 있습니다. 기존 정책 그룹은 IOPS 기준으로 최대(또는 최신 버전에서는 최소) 처리량을 고정적으로 제공합니다. 적응형 QoS는 워크로드 크기에 따라 처리량을 자동으로 확장하여 워크로드 크기 변화에 따른 IOPS 대 TB/GB 비율을 유지합니다. 이는 대규모 배포 환경에서 수백 또는 수천 개의 워크로드를 관리할 때 상당한 이점을 제공합니다.

QoS 정책 그룹을 생성할 때 다음 사항을 고려하십시오.

- 백엔드 구성의 `qosPolicy` 블록에서 `defaults` 키를 설정해야 합니다. 다음은 백엔드 구성 예시입니다.

```
---
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
```

- 볼륨당 정책 그룹을 적용해야 각 볼륨이 정책 그룹에서 지정한 전체 처리량을 얻을 수 있습니다. 공유 정책 그룹은 지원되지 않습니다.

QoS 정책 그룹에 대한 자세한 내용은 "ONTAP 명령 참조"를 참조하십시오.

Kubernetes 클러스터 구성원으로 스토리지 리소스 액세스 제한

Trident에서 생성한 NFS 볼륨, iSCSI LUN 및 FC LUN에 대한 액세스를 제한하는 것은 Kubernetes 배포 환경의 보안 태세에서 중요한 구성 요소입니다. 이를 통해 Kubernetes 클러스터에 속하지 않은 호스트가 해당 볼륨에 액세스하여 예기치 않게 데이터를 수정하는 것을 방지할 수 있습니다.

Kubernetes에서 네임스페이스는 리소스의 논리적 경계라는 점을 이해하는 것이 중요합니다. 동일한 네임스페이스 내의 리소스는 공유될 수 있다는 가정이 있지만, 중요한 것은 네임스페이스 간 기능이 없다는 것입니다. 즉, PV는 전역 객체이지만 PVC에 바인딩되면 동일한 네임스페이스에 있는 Pod에서만 액세스할 수 있습니다. 적절한 경우 네임스페이스를 사용하여 분리를 제공하는 것이 중요합니다.

Kubernetes 환경에서 데이터 보안과 관련하여 대부분의 조직이 가장 우려하는 사항은 컨테이너의 프로세스가 호스트에 마운트된 스토리지에 액세스할 수 있지만 컨테이너용이 아니라는 점입니다. "네임스페이스"는 이러한 유형의 침해를 방지하도록 설계되었습니다. 그러나 예외가 하나 있습니다. 바로 권한 있는 컨테이너입니다.

특권 컨테이너는 일반 컨테이너보다 훨씬 더 많은 호스트 수준 권한으로 실행되는 컨테이너입니다. 이러한 권한은 기본적으로 거부되지 않으므로 "Pod 보안 정책"을 사용하여 해당 기능을 비활성화해야 합니다.

Kubernetes와 외부 호스트 모두에서 액세스가 필요한 볼륨의 경우, 스토리지는 기존 방식으로 관리되어야 하며, PV는 관리자가 도입하고 Trident에서 관리하지 않아야 합니다. 이렇게 하면 Kubernetes와 외부 호스트 모두 연결이 끊어지고 더 이상 볼륨을 사용하지 않을 때만 스토리지 볼륨이 삭제됩니다. 또한, 사용자 지정 내보내기 정책을 적용하여 Kubernetes 클러스터 노드와 Kubernetes 클러스터 외부의 대상 서버에서 액세스할 수 있도록 할 수 있습니다.

전용 인프라 노드(예: OpenShift) 또는 사용자 애플리케이션 예약이 불가능한 노드가 있는 배포 환경의 경우, 스토리지

리소스에 대한 액세스를 더욱 제한하기 위해 별도의 내보내기 정책을 사용해야 합니다. 여기에는 해당 인프라 노드에 배포된 서비스(예: OpenShift 메트릭 및 로깅 서비스)와 인프라 노드가 아닌 곳에 배포된 표준 애플리케이션에 대한 내보내기 정책을 생성하는 것이 포함됩니다.

전용 엑스포트 정책 사용

각 백엔드에 대해 Kubernetes 클러스터에 있는 노드에만 액세스를 허용하는 엑스포트 정책이 있는지 확인해야 합니다. Trident는 엑스포트 정책을 자동으로 생성하고 관리할 수 있습니다. 이를 통해 Trident는 프로비저닝하는 볼륨에 대한 액세스를 Kubernetes 클러스터의 노드로 제한하고 노드 추가/삭제를 간소화합니다.

또는 수동으로 엑스포트 정책을 생성하고 각 노드 액세스 요청을 처리하는 하나 이상의 엑스포트 규칙을 추가할 수도 있습니다.

- `vserver export-policy create` ONTAP CLI 명령을 사용하여 엑스포트 정책을 생성합니다.
- `vserver export-policy rule create` ONTAP CLI 명령을 사용하여 엑스포트 정책에 규칙을 추가합니다.

이러한 명령을 실행하면 데이터에 액세스할 수 있는 Kubernetes 노드를 제한할 수 있습니다.

`showmount`애플리케이션 SVM에 대해 비활성화합니다

`showmount` 기능을 사용하면 `NFS` 클라이언트가 `SVM`에 사용 가능한 `NFS` 내보내기 목록을 쿼리할 수 있습니다. `Kubernetes` 클러스터에 배포된 `Pod`는 `showmount -e` 명령을 실행하여 액세스 권한이 없는 마운트를 포함하여 사용 가능한 마운트 목록을 받을 수 있습니다. 이는 그 자체로는 보안 침해가 아니지만, 권한이 없는 사용자가 `NFS` 내보내기에 연결하는 데 도움이 될 수 있는 불필요한 정보를 제공합니다.

SVM 레벨 ONTAP CLI 명령을 사용하여 `showmount` 비활성화해야 합니다.

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 모범 사례

Trident용 SolidFire 스토리지 구성 모범 사례를 알아보십시오.

SolidFire 계정 생성

각 SolidFire 계정은 고유한 볼륨 소유자를 나타내며 고유한 CHAP(Challenge-Handshake Authentication Protocol) 자격 증명 세트를 받습니다. 계정에 할당된 볼륨은 계정 이름과 관련 CHAP 자격 증명을 사용하거나 볼륨 액세스 그룹을 통해 액세스할 수 있습니다. 계정에는 최대 2,000개의 볼륨을 할당할 수 있지만 볼륨은 하나의 계정에만 속할 수 있습니다.

QoS 정책을 생성합니다

여러 볼륨에 적용할 수 있는 표준화된 서비스 품질 설정을 생성하고 저장하려면 SolidFire QoS(Quality of Service) 정책을 사용하십시오.

볼륨별로 QoS 파라미터를 설정할 수 있습니다. QoS를 정의하는 세 가지 구성 가능한 파라미터(최소 IOPS, 최대 IOPS,

버스트 IOPS)를 설정하여 각 볼륨의 성능을 보장할 수 있습니다.

다음은 4Kb 블록 크기에 대한 최소, 최대 및 버스트 IOPS 값입니다.

IOPS 매개변수	정의	최소값	기본값	최대값**(4Kb)**
최소 IOPS	볼륨에 대해 보장되는 성능 수준입니다.	50	50	15000
최대 IOPS	성능은 이 한도를 초과하지 않습니다.	50	15000	200,000
버스트 IOPS	단기 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 노드별 성능에 따라 제한됩니다.

블록 크기와 대역폭은 IOPS 수에 직접적인 영향을 미칩니다. 블록 크기가 증가함에 따라 시스템은 더 큰 블록 크기를 처리하는 데 필요한 수준까지 대역폭을 증가시킵니다. 대역폭이 증가하면 시스템이 달성할 수 있는 IOPS 수는 감소합니다. QoS 및 성능에 대한 자세한 내용은 "SolidFire 서비스 품질"을 참조하십시오.

SolidFire 인증

Element는 CHAP와 VAG(Volume Access Groups)라는 두 가지 인증 방식을 지원합니다. CHAP는 CHAP 프로토콜을 사용하여 호스트를 백엔드에 인증합니다. Volume Access Groups는 프로비저닝하는 볼륨에 대한 액세스를 제어합니다. NetApp은 CHAP 인증 방식이 더 간단하고 확장성에 제한이 없으므로 CHAP 사용을 권장합니다.

향상된 CSI 프로비저너를 사용하는 Trident는 CHAP 인증을 지원합니다. VAG는 기존의 비CSI 운영 모드에서만 사용해야 합니다.

CHAP 인증(시작자가 의도된 볼륨 사용자인지 확인하는 인증)은 계정 기반 액세스 제어에서만 지원됩니다. CHAP를 사용하여 인증하는 경우 단방향 CHAP와 양방향 CHAP의 두 가지 옵션을 사용할 수 있습니다. 단방향 CHAP는 SolidFire 계정 이름과 시작자 암호를 사용하여 볼륨 액세스를 인증합니다. 양방향 CHAP 옵션은 볼륨이 계정 이름과 시작자 암호를 통해 호스트를 인증하고, 호스트는 계정 이름과 대상 암호를 통해 볼륨을 인증하는 가장 안전한 방법입니다.

하지만 CHAP를 활성화할 수 없고 VAG가 필요한 경우 액세스 그룹을 생성하고 호스트 이니시에이터와 볼륨을 액세스 그룹에 추가하십시오. 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 Volume Access Group 액세스 제어가 사용됩니다.

자세한 정보는 어디에서 찾을 수 있습니까?

다음은 모범 사례 문서의 일부 목록입니다. "NetApp 라이브러리"에서 최신 버전을 검색하십시오.

ONTAP

- "NFS 모범 사례 및 구현 가이드"
- "SAN 관리" (iSCSI용)
- "RHEL용 iSCSI Express 구성"

Element 소프트웨어

- "Linux용 SolidFire 구성"

NetApp HCI

- "NetApp HCI 배포 필수 조건"
- "NetApp Deployment Engine에 액세스합니다"

애플리케이션 모범 사례 정보

- "ONTAP의 MySQL 모범 사례"
- "SolidFire의 MySQL 모범 사례"
- "NetApp SolidFire 및 Cassandra"
- "SolidFire의 Oracle 모범 사례"
- "SolidFire의 PostgreSQL 모범 사례"

모든 애플리케이션에 특정 지침이 있는 것은 아니므로 NetApp 팀과 협력하고 "NetApp 라이브러리"를 사용하여 최신 문서를 찾는 것이 중요합니다.

Trident 통합

Trident를 통합하려면 드라이버 선택 및 배포, 스토리지 클래스 설계, 가상 풀 설계, 스토리지 프로비저닝에 대한 영구 볼륨 클레임(PVC) 영향, 볼륨 작업 및 Trident를 사용한 OpenShift 서비스 배포와 같은 설계 및 아키텍처 요소를 통합해야 합니다.

드라이버 선택 및 배포

스토리지 시스템의 백엔드 드라이버를 선택하고 배포합니다.

ONTAP 백엔드 드라이버

ONTAP 백엔드 드라이버는 사용되는 프로토콜과 스토리지 시스템에서 볼륨이 프로비저닝되는 방식에 따라 구분됩니다. 따라서 어떤 드라이버를 배포할지 결정할 때 신중하게 고려해야 합니다.

좀 더 자세히 설명하자면, 애플리케이션에 공유 스토리지(여러 파드가 동일한 PVC에 액세스하는 경우)가 필요한 구성 요소가 있다면 NAS 기반 드라이버가 기본 선택 사항이 되고, 블록 기반 iSCSI 드라이버는 공유 스토리지가 필요 없는 경우에 적합합니다. 애플리케이션 요구 사항과 스토리지 및 인프라 팀의 숙련도를 고려하여 프로토콜을 선택하십시오. 일반적으로 대부분의 애플리케이션에서 두 프로토콜 간의 차이는 크지 않으므로, 여러 파드가 동시에 액세스해야 하는 공유 스토리지가 필요한지 여부에 따라 결정되는 경우가 많습니다.

사용 가능한 ONTAP 백엔드 드라이버는 다음과 같습니다.

- `ontap-nas`: 프로비저닝된 각 PV는 완전한 ONTAP FlexVolume입니다.
- `ontap-nas-economy`: 프로비저닝된 각 PV는 qtree이며, FlexVolume당 구성 가능한 qtree 수를 가집니다(기본값은 200개).
- `ontap-nas-flexgroup`: 각 PV는 전체 ONTAP FlexGroup로 프로비저닝되며 SVM에 할당된 모든 애그리게이트가 사용됩니다.
- `ontap-san`: 프로비저닝된 각 PV는 자체 FlexVolume 내의 LUN입니다.
- `ontap-san-economy`: 프로비저닝된 각 PV는 LUN이며, FlexVolume당 구성 가능한 LUN 수가 있습니다(기본값은 100개).

세 가지 NAS 드라이버 중에서 선택하는 것은 애플리케이션에서 사용할 수 있는 기능에 몇 가지 영향을 미칩니다.

아래 표에서 모든 기능이 Trident를 통해 노출되는 것은 아닙니다. 일부 기능은 필요한 경우 스토리지 관리자가 프로비저닝 후 적용해야 합니다. 위첨자 각주는 기능 및 드라이버별 기능을 구분하여 보여줍니다.

ONTAP NAS 드라이버	스냅샷	클론	동적 엑스포트 정책	다중 연결	**QoS**	크기 조정	복제
`ontap-nas`	예	예	예 [5]	예	예 [1]	예	예 [1]
`ontap-nas-economy`	NO [3]	NO [3]	예 [5]	예	NO [3]	예	NO [3]
`ontap-nas-flexgroup`	예 [1]	아니요	예 [5]	예	예 [1]	예	예 [1]

Trident는 ONTAP용 SAN 드라이버 2개를 제공하며, 해당 기능은 아래와 같습니다.

ONTAP SAN 드라이버	스냅샷	클론	다중 연결	양방향 **CHAP**	**QoS**	크기 조정	복제
`ontap-san`	예	예	예 [4]	예	예 [1]	예	예 [1]
`ontap-san-economy`	예	예	예 [4]	예	NO [3]	예	NO [3]

> 위 표에 대한 각주: 예각주:1[]: Trident에서 관리하지 않음 예각주:2[]: Trident에서 관리하지만 PV 세분화는 지원하지 않음 아니오각주:3[]: Trident에서 관리하지 않으며 PV 세분화도 지원하지 않음 예각주:4[]: 원시 블록 볼륨에 대해 지원됨 예각주:5[]: Trident에서 지원됨

PV 단위로 세분화되지 않은 기능은 전체 FlexVolume에 적용되며 모든 PV(즉, 공유 FlexVols의 qtree 또는 LUN)는 공통 스케줄을 공유합니다.

위 표에서 볼 수 있듯이 `ontap-nas`와 `ontap-nas-economy`의 대부분의 기능은 동일합니다. 그러나 `ontap-nas-economy` 드라이버는 PV 단위로 스케줄을 제어하는 기능을 제한하기 때문에 특히 재해 복구 및 백업 계획에 영향을 미칠 수 있습니다. ONTAP 스토리지에서 PVC 클론 기능을 활용하려는 개발 팀의 경우 `ontap-nas`, `ontap-san` 또는 `ontap-san-economy` 드라이버를 사용할 때만 가능합니다.

> `solidfire-san` 드라이버는 PVC를 복제하는 기능도 갖추고 있습니다.

Cloud Volumes ONTAP 백엔드 드라이버

Cloud Volumes ONTAP는 파일 공유 및 NAS 및 SAN 프로토콜(NFS, SMB/CIFS 및 iSCSI)을 지원하는 블록 레벨 스토리지를 포함한 다양한 사용 사례를 위한 엔터프라이즈급 스토리지 기능과 함께 데이터 제어 기능을 제공합니다. Cloud Volume ONTAP와 호환되는 드라이버는 `ontap-nas`, `ontap-nas-economy`, `ontap-san` 및 `ontap-san-economy`입니다. 이러한 드라이버는 Azure용 Cloud Volume ONTAP, GCP용 Cloud Volume ONTAP에 적용됩니다.

Amazon FSx for ONTAP 백엔드 드라이버

Amazon FSx for NetApp ONTAP을 사용하면 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하는 동시에 AWS에 데이터를 저장하는 간편성, 민첩성, 보안 및 확장성의 이점을 누릴 수 있습니다. FSx for ONTAP은 다양한 ONTAP 파일 시스템 기능과 관리 API를 지원합니다. Cloud Volume ONTAP과 호환되는 드라이버는 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` 및 `ontap-san-economy`입니다.

NetApp HCI/SolidFire 백엔드 드라이버

`solidfire-san` NetApp HCI/SolidFire 플랫폼에서 사용되는 드라이버를 통해 관리자는 QoS 제한을 기반으로 Trident용 Element 백엔드를 구성할 수 있습니다. Trident에서 프로비저닝한 볼륨에 특정 QoS 제한을 설정하도록 백엔드를 설계하려면 백엔드 파일에서 `type` 매개변수를 사용하십시오. 관리자는 또한 `limitVolumeSize` 매개변수를 사용하여 스토리지에 생성할 수 있는 볼륨 크기를 제한할 수 있습니다. 현재 볼륨 크기 조정 및 볼륨 복제와 같은 Element 스토리지 기능은 `solidfire-san` 드라이버를 통해 지원되지 않습니다. 이러한 작업은 Element Software 웹 UI를 통해 수동으로 수행해야 합니다.

SolidFire 드라이버	스냅샷	클론	다중 연결	CHAP	QoS	크기 조정	복제
solidfire-san	예	예	예 [2]	예	예	예	예 [1]

각주: 예각주:1[]: Trident에서 관리하지 않음 예각주:2[]: 원시 블록 볼륨에 대해 지원됨

Azure NetApp Files 백엔드 드라이버

Trident는 `azure-netapp-files` 드라이버를 사용하여 "Azure NetApp Files" 서비스를 관리합니다.

이 드라이버 및 구성 방법에 대한 자세한 내용은 "Azure NetApp Files용 Trident 백엔드 구성"에서 확인할 수 있습니다.

Azure NetApp Files 드라이버	스냅샷	클론	다중 연결	QoS	확장	복제
azure-netapp-files	예	예	예	예	예	예 [1]

각주: 예각주:1[]: Trident에서 관리되지 않음

스토리지 클래스 설계

Kubernetes 스토리지 클래스 객체를 생성하려면 개별 스토리지 클래스를 구성하고 적용해야 합니다. 이 섹션에서는 애플리케이션에 맞는 스토리지 클래스를 설계하는 방법을 설명합니다.

특정 백엔드 활용

특정 스토리지 클래스 객체 내에서 필터링을 사용하여 해당 스토리지 클래스와 함께 사용할 스토리지 풀 또는 풀 집합을 결정할 수 있습니다. 스토리지 클래스에는 세 가지 필터 집합 `storagePools`, `additionalStoragePools` 및 /또는 `excludeStoragePools`을 설정할 수 있습니다.

`storagePools` 매개변수는 지정된 속성과 일치하는 풀 집합으로 스토리지를 제한하는 데 도움이 됩니다. `additionalStoragePools` 매개변수는 속성 및 `storagePools` 매개변수로 선택된 풀 집합과 함께 Trident가 프로비저닝에 사용하는 풀 집합을 확장하는 데 사용됩니다. 적절한 스토리지 풀 집합이 선택되도록 하려면 두 매개변수를 단독으로 또는 함께 사용할 수 있습니다.

`excludeStoragePools` 매개변수는 속성과 일치하는 나열된 풀 집합을 명시적으로 제외하는 데 사용됩니다.

QoS 정책 에뮬레이트

QoS(서비스 품질) 정책을 모방하도록 스토리지 클래스를 설계하려면 `media` 속성을 `hdd` 또는 `ssd`로 설정하여 스토리지 클래스를 생성하십시오. 스토리지 클래스에 지정된 `media` 속성을 기반으로 Trident는 `hdd` 또는 `ssd` 애그리게이트를 제공하는 적절한 백엔드를 선택하여 미디어 속성과 일치시키고 특정 애그리게이트에 볼륨 프로비저닝을 지시합니다. 따라서 `media` 속성을 `ssd`로 설정한 PREMIUM 스토리지 클래스를 생성하면 PREMIUM QoS 정책으로 분류할 수 있습니다. 마찬가지로 미디어 속성을 `hdd'로 설정한 STANDARD 스토리지 클래스를 생성하면 STANDARD QoS 정책으로 분류할 수 있습니다. 또한 스토리지 클래스의 ``IOPS" 속성을 사용하여 Element 어플라이언스로 프로비저닝을 리디렉션하는 것도 QoS 정책으로 정의할 수 있습니다.

특정 기능을 기반으로 백엔드 활용

스토리지 클래스는 씬 프로비저닝 및 씩 프로비저닝, 스냅샷, 클론, 암호화와 같은 기능이 활성화된 특정 백엔드에서 볼륨 프로비저닝을 수행하도록 설계할 수 있습니다. 사용할 스토리지를 지정하려면 필요한 기능이 활성화된 적절한 백엔드를 지정하는 스토리지 클래스를 생성하십시오.

가상 풀

가상 풀은 모든 Trident 백엔드에서 사용할 수 있습니다. Trident에서 제공하는 모든 드라이버를 사용하여 모든 백엔드에 대한 가상 풀을 정의할 수 있습니다.

가상 풀을 사용하면 관리자가 스토리지 클래스를 통해 참조할 수 있는 백엔드에 대한 추상화 계층을 생성하여 백엔드에 볼륨을 더욱 유연하고 효율적으로 배치할 수 있습니다. 동일한 서비스 클래스로 여러 백엔드를 정의할 수 있습니다. 또한 동일한 백엔드에 서로 다른 특성을 가진 여러 스토리지 풀을 생성할 수도 있습니다. 스토리지 클래스에 특정 레이블이 있는 선택기가 구성되면 Trident는 선택기의 모든 레이블과 일치하는 백엔드를 선택하여 볼륨을 배치합니다. 스토리지 클래스 선택기 레이블이 여러 스토리지 풀과 일치하는 경우 Trident는 그중 하나를 선택하여 볼륨을 프로비저닝합니다.

가상 풀 설계

백엔드를 생성할 때 일반적으로 매개변수 세트를 지정할 수 있습니다. 관리자가 동일한 스토리지 자격 증명을 사용하면서 다른 매개변수 세트를 지정하여 다른 백엔드를 생성하는 것이 불가능했습니다. 가상 풀이 도입되면서 이 문제가 해결되었습니다. 가상 풀은 백엔드와 Kubernetes 스토리지 클래스 사이에 도입된 추상화 계층으로, 관리자가 백엔드에 구애받지 않고 Kubernetes 스토리지 클래스에서 선택기로 참조할 수 있는 레이블과 함께 매개변수를 정의할

수 있도록 합니다. 가상 풀은 Trident에서 지원되는 모든 NetApp 백엔드에 대해 정의할 수 있습니다. 여기에는 SolidFire/NetApp HCI, ONTAP 및 Azure NetApp Files가 포함됩니다.

> 가상 풀을 정의할 때 백엔드 정의에서 기존 가상 풀의 순서를 변경하지 않는 것이 좋습니다. 또한 기존 가상 풀의 속성을 편집/수정하지 말고 새 가상 풀을 정의하는 것이 좋습니다.

다양한 서비스 수준/**QoS** 에뮬레이션

서비스 클래스를 에뮬레이션하기 위한 가상 풀을 설계할 수 있습니다. Azure NetApp Files용 Cloud Volume Service의 가상 풀 구현을 사용하여 다양한 서비스 클래스를 설정하는 방법을 살펴보겠습니다. Azure NetApp Files 백엔드에 서로 다른 성능 수준을 나타내는 여러 레이블을 구성합니다. `servicelevel` 애스펙트를 적절한 성능 수준으로 설정하고 각 레이블 아래에 필요한 다른 애스펙트를 추가합니다. 이제 서로 다른 가상 풀에 매핑되는 다양한 Kubernetes 스토리지 클래스를 생성합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다.

특정 측면 집합 할당

하나의 스토리지 백엔드에서 특정 속성을 가진 여러 개의 가상 풀을 설계할 수 있습니다. 이를 위해 백엔드에 여러 레이블을 구성하고 각 레이블 아래에 필요한 속성을 설정합니다. 이제 `parameters.selector` 필드를 사용하여 각 가상 풀에 매핑되는 서로 다른 Kubernetes 스토리지 클래스를 생성합니다. 백엔드에 프로비저닝되는 볼륨은 선택한 가상 풀에 정의된 속성을 갖게 됩니다.

스토리지 프로비저닝에 영향을 미치는 **PVC** 특성

PVC를 생성할 때 요청된 스토리지 클래스 외의 일부 매개변수가 Trident 프로비저닝 결정 프로세스에 영향을 미칠 수 있습니다.

액세스 모드

PVC를 통해 스토리지를 요청할 때 필수 입력 항목 중 하나는 액세스 모드입니다. 원하는 모드에 따라 스토리지 요청을 호스팅하는 백엔드가 선택될 수 있습니다.

Trident는 다음 매트릭스에 따라 지정된 액세스 방법과 함께 사용되는 스토리지 프로토콜을 일치시키려고 시도합니다. 이는 기본 스토리지 플랫폼과 무관합니다.

	ReadWriteOnce	**ReadOnlyMany**	**ReadWriteMany**
iSCSI	예	예	예(Raw 블록)
NFS	예	예	예

NFS 백엔드가 구성되지 않은 Trident 배포 환경에 ReadWriteMany PVC를 요청하면 볼륨이 프로비저닝되지 않습니다. 따라서 요청자는 애플리케이션에 적합한 액세스 모드를 사용해야 합니다.

볼륨 작업

영구 볼륨 수정

Kubernetes에서 영구 볼륨은 두 가지 예외를 제외하고는 변경 불가능한 객체입니다. 일단 생성되면 회수 정책과 크기는 수정할 수 있습니다. 하지만 그렇다고 해서 Kubernetes 외부에서 볼륨의 일부 측면을 수정하는 것이 불가능한 것은 아닙니다. 특정 애플리케이션에 맞게 볼륨을 사용자 지정하거나, 용량이 실수로 소모되는 것을 방지하거나, 또는 어떤 이유로든 볼륨을 다른 스토리지 컨트롤러로 이동해야 하는 경우에 이러한 수정이 필요할 수 있습니다.

현재 Kubernetes 내장 프로비저너는 NFS, iSCSI 또는 FC PV에 대한 볼륨 크기 조정 작업을 지원하지 않습니다. Trident는 NFS, iSCSI 및 FC 볼륨 모두의 확장을 지원합니다.

PV의 연결 세부 정보는 생성 후 수정할 수 없습니다.

필요 시 볼륨 스냅샷 생성

Trident는 CSI 프레임워크를 사용하여 온디맨드 볼륨 스냅샷 생성 및 스냅샷으로부터 PVC 생성을 지원합니다. 스냅샷은 데이터의 특정 시점 복사본을 유지하는 편리한 방법을 제공하며 Kubernetes의 소스 PV와는 독립적인 수명 주기를 가집니다. 이러한 스냅샷을 사용하여 PVC를 복제할 수 있습니다.

스냅샷에서 볼륨 생성

Trident는 볼륨 스냅샷에서 PersistentVolumes를 생성하는 기능도 지원합니다. 이를 위해 PersistentVolumeClaim을 생성하고 `datasource`볼륨을 생성할 필요한 스냅샷으로 지정하기만 하면 됩니다. Trident는 스냅샷에 있는 데이터로 볼륨을 생성하여 이 PVC를 처리합니다. 이 기능을 사용하면 리전 간 데이터 복제, 테스트 환경 생성, 손상되거나 오류가 발생한 프로덕션 볼륨 전체 교체 또는 특정 파일 및 디렉토리 검색 후 다른 연결된 볼륨으로 전송이 가능합니다.

클러스터에서 볼륨 이동

스토리지 관리자는 ONTAP 클러스터 내의 애그리게이트와 컨트롤러 간에 스토리지 소비자에게 중단 없이 볼륨을 이동할 수 있습니다. 대상 애그리게이트가 Trident에서 사용 중인 SVM에 접근 권한이 있는 애그리게이트인 한 이 작업은 Trident 또는 Kubernetes 클러스터에 영향을 미치지 않습니다. 중요한 점은 애그리게이트가 SVM에 새로 추가된 경우 Trident에 다시 추가하여 백엔드를 새로 고쳐야 한다는 것입니다. 이렇게 하면 Trident가 SVM의 인벤토리를 다시 생성하여 새 애그리게이트를 인식하게 됩니다.

하지만 백엔드 간에 볼륨을 이동하는 것은 Trident에서 자동으로 지원되지 않습니다. 여기에는 동일한 클러스터 내의 SVM 간, 클러스터 간 또는 다른 스토리지 플랫폼으로의 이동(해당 스토리지 시스템이 Trident에 연결되어 있는 경우에도)이 포함됩니다.

볼륨을 다른 위치로 복사한 경우, 볼륨 가져오기 기능을 사용하여 현재 볼륨을 Trident로 가져올 수 있습니다.

볼륨 확장

Trident는 NFS, iSCSI, 그리고 FC PV의 크기 조정을 지원합니다. 이를 통해 사용자는 Kubernetes 계층에서 직접 볼륨의 크기를 조정할 수 있습니다. 볼륨 확장은 ONTAP를 포함한 모든 주요 NetApp 스토리지 플랫폼과 SolidFire/NetApp HCI 백엔드에서 가능합니다. 나중에 확장이 가능하도록, 볼륨과 연결된 StorageClass에서 `allowVolumeExpansion`를 `true`로 설정하세요. `Persistent Volume`의 크기를 조정해야 할 때마다 `Persistent Volume Claim`의 `spec.resources.requests.storage` 주석을 원하는 볼륨 크기로 수정하세요. Trident가 스토리지 클러스터에서 볼륨 크기 조정을 자동으로 처리합니다.

기존 볼륨을 **Kubernetes**로 가져오기

볼륨 가져오기 기능을 사용하면 기존 스토리지 볼륨을 Kubernetes 환경으로 가져올 수 있습니다. 현재 이 기능은 `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` 및 `azure-netapp-files` 드라이버에서 지원됩니다. 이 기능은 기존 애플리케이션을 Kubernetes로 포팅하거나 재해 복구 시나리오에서 유용합니다.

ONTAP 및 `solidfire-san`드라이버를 사용하는 경우 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`명령을 사용하여 기존 볼륨을 Kubernetes로 가져와 Trident에서 관리할 수 있습니다. 볼륨 가져오기 명령에 사용되는 PVC YAML 또는 JSON 파일은 Trident를 프로비저너로 식별하는 스토리지 클래스를 가리킵니다. NetApp HCI/SolidFire 백엔드를 사용하는 경우 볼륨 이름이 고유한지 확인하십시오. 볼륨 이름이 중복되는 경우 볼륨을 고유한 이름으로 복제하여 볼륨 가져오기 기능이 구분할 수 있도록 하십시오.

`azure-netapp-files` 드라이버를 사용하는 경우 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 명령을 사용하여 볼륨을 Kubernetes로 가져와 Trident에서 관리합니다. 이렇게 하면 고유한 볼륨 참조가 보장됩니다.

위 명령이 실행되면 Trident는 백엔드에서 볼륨을 찾아 크기를 읽습니다. 구성된 PVC의 볼륨 크기를 자동으로 추가하고 필요한 경우 덮어씁니다. 그런 다음 Trident는 새 PV를 생성하고 Kubernetes는 PVC를 PV에 바인딩합니다.

특정 가져온 PVC가 필요한 방식으로 컨테이너가 배포된 경우, 볼륨 가져오기 프로세스를 통해 PVC/PV 쌍이 바인딩될 때까지 컨테이너는 대기 상태로 유지됩니다. PVC/PV 쌍이 바인딩되면 다른 문제가 없는 한 컨테이너가 정상적으로 실행됩니다.

레지스트리 서비스

레지스트리용 스토리지 배포 및 관리에 대한 내용은 "netapp.io"의 "블로그"에 설명되어 있습니다.

로깅 서비스

다른 OpenShift 서비스와 마찬가지로 로깅 서비스는 플레이북에 제공되는 인벤토리 파일(일명 hosts)에 포함된 구성 매개변수를 사용하여 Ansible로 배포됩니다. 두 가지 설치 방법이 다루어집니다. 초기 OpenShift 설치 중 로깅 배포와 OpenShift 설치 후 로깅 배포입니다.

> Red Hat OpenShift 버전 3.9부터 공식 문서에서는 데이터 손상 문제로 인해 로깅 서비스에 NFS를 사용하지 않을 것을 권장합니다. 이는 Red Hat의 제품 테스트 결과를 기반으로 합니다. ONTAP NFS 서버에는 이러한 문제가 없으며 로깅 배포를 원활하게 지원할 수 있습니다. 궁극적으로 로깅 서비스에 사용할 프로토콜 선택은 사용자에게 달려 있지만, NetApp 플랫폼을 사용하는 경우 두 프로토콜 모두 훌륭하게 작동하며 NFS를 선호하는 경우 이를 피할 이유가 없습니다.

로깅 서비스에 NFS를 사용하려면 Ansible 변수 `openshift_enable_unsupported_configurations`를 `true`로 설정하여 설치 프로그램 오류를 방지해야 합니다.

시작하기

로깅 서비스는 선택적으로 애플리케이션뿐 아니라 OpenShift 클러스터 자체의 핵심 운영에도 배포할 수 있습니다. 변수 `openshift_logging_use_ops`를 `true`로 지정하여 운영 로깅을 배포하도록 선택하면 서비스 인스턴스가 두 개 생성됩니다. 운영 로깅 인스턴스를 제어하는 변수에는 "ops"가 포함되고, 애플리케이션용 인스턴스에는 포함되지 않습니다.

배포 방법에 따라 Ansible 변수를 구성하는 것은 기본 서비스에서 올바른 스토리지를 사용하도록 보장하는 데 중요합니다. 각 배포 방법에 대한 옵션을 살펴보겠습니다.

> 아래 표에는 로깅 서비스와 관련된 스토리지 구성에 필요한 변수만 포함되어 있습니다. "Red Hat OpenShift 로깅 문서"에서 배포 환경에 따라 검토, 구성 및 사용해야 하는 다른 옵션을 찾을 수 있습니다.

아래 표의 변수를 사용하면 Ansible 플레이북이 제공된 세부 정보를 사용하여 로깅 서비스용 PV 및 PVC를 생성합니다. 이 방법은 OpenShift 설치 후 구성 요소 설치 플레이북을 사용하는 것보다 유연성이 떨어지지만, 기존 볼륨이 있는 경우 하나의 옵션으로 사용할 수 있습니다.

변수	세부 정보
`openshift_logging_storage_kind`	`nfs`로 설정하여 설치 프로그램이 로깅 서비스를 위한 NFS PV를 생성하도록 합니다.
`openshift_logging_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이 값은 가상 머신의 dataLIF로 설정해야 합니다.
`openshift_logging_storage_nfs_directory`	NFS 내보내기의 마운트 경로입니다. 예를 들어 볼륨이 `/openshift_logging`로 접합된 경우 해당 경로를 이 변수에 사용합니다.
`openshift_logging_storage_volume_name`	생성할 PV의 이름(예: `pv_ose_logs`).
`openshift_logging_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi`.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
`openshift_logging_es_pvc_dynamic`	동적으로 프로비저닝된 볼륨을 사용하려면 true로 설정하십시오.
`openshift_logging_es_pvc_storage_class_name`	PVC에서 사용될 스토리지 클래스의 이름입니다.
`openshift_logging_es_pvc_size`	PVC에서 요청한 볼륨의 크기입니다.
`openshift_logging_es_pvc_prefix`	로깅 서비스에서 사용하는 PVC의 접두사입니다.
`openshift_logging_es_ops_pvc_dynamic`	`true`로 설정하여 운영 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 PVC의 접두사입니다.

로깅 스택을 배포합니다

초기 OpenShift 설치 과정의 일부로 로깅을 배포하는 경우 표준 배포 프로세스만 따르면 됩니다. Ansible은 필요한 서비스와 OpenShift 객체를 구성하고 배포하므로 Ansible 작업이 완료되는 즉시 서비스를 사용할 수 있습니다.

하지만 초기 설치 후 배포하는 경우에는 Ansible에서 컴포넌트 플레이북을 사용해야 합니다. 이 과정은 OpenShift 버전에 따라 약간 다를 수 있으므로 사용 중인 버전에 맞는 "Red Hat OpenShift Container Platform 3.11 문서"를 읽고 따르십시오.

메트릭 서비스

메트릭 서비스는 관리자에게 OpenShift 클러스터의 상태, 리소스 활용률 및 가용성에 대한 유용한 정보를 제공합니다. 또한 Pod 자동 스케일링 기능에 필수적이며, 많은 조직에서 메트릭 서비스의 데이터를 비용 청구 및/또는 사용량 보고 애플리케이션에 활용합니다.

로깅 서비스 및 OpenShift 전체와 마찬가지로 Ansible은 메트릭 서비스를 배포하는 데 사용됩니다. 또한 로깅 서비스와

마찬가지로 메트릭 서비스는 클러스터 초기 설정 시 또는 운영 후 구성 요소 설치 방법을 사용하여 배포할 수 있습니다. 다음 표에는 메트릭 서비스의 영구 스토리지를 구성할 때 중요한 변수가 나와 있습니다.

아래 표에는 메트릭 서비스와 관련된 스토리지 구성에 필요한 변수만 포함되어 있습니다. 문서에는 이 외에도 다양한 옵션이 있으므로 배포 환경에 맞게 검토, 구성 및 사용해야 합니다.

변수	세부 정보
`openshift_metrics_storage_kind`	`nfs`로 설정하여 설치 프로그램이 로깅 서비스를 위한 NFS PV를 생성하도록 합니다.
`openshift_metrics_storage_host`	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 이 값은 SVM의 dataLIF로 설정해야 합니다.
`openshift_metrics_storage_nfs_directory`	NFS 내보내기의 마운트 경로입니다. 예를 들어 볼륨이 `/openshift_metrics`로 접합된 경우 해당 경로를 이 변수에 사용합니다.
`openshift_metrics_storage_volume_name`	생성할 PV의 이름(예: `pv_ose_metrics`).
`openshift_metrics_storage_volume_size`	예를 들어 NFS 내보내기의 크기 `100Gi`.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 프로그램은 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
`openshift_metrics_cassandra_pvc_prefix`	메트릭 PVC에 사용할 접두사입니다.
`openshift_metrics_cassandra_pvc_size`	요청할 볼륨의 크기입니다.
`openshift_metrics_cassandra_storage_type`	메트릭에 사용할 스토리지 유형입니다. Ansible이 적절한 스토리지 클래스를 가진 PVC를 생성하려면 이 값을 dynamic으로 설정해야 합니다.
`openshift_metrics_cassanda_pvc_storage_class_name`	사용할 스토리지 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. 컴포넌트 플레이북을 사용하여 OpenShift 설치 후 배포하는 경우 Ansible이 필요한 모든 PVC를 생성하고 Trident가 해당 PVC에 대한 스토리지를 프로비저닝한 후 서비스를 배포합니다.

위의 변수 및 배포 프로세스는 OpenShift의 각 버전에 따라 변경될 수 있습니다. 사용 중인 버전에 맞는 "Red Hat의 OpenShift 배포 가이드"을(를) 검토하고 따라 환경에 맞게 구성되도록 하십시오.

데이터 보호 및 재해 복구

Trident 및 Trident를 사용하여 생성된 볼륨에 대한 보호 및 복구 옵션에 대해 알아보십시오. 영구 저장 요구 사항이 있는 각 애플리케이션에 대해 데이터 보호 및 복구 전략을 수립해야 합니다.

Trident 복제 및 복구

재해 발생 시 Trident를 복원하기 위해 백업을 생성할 수 있습니다.

Trident 복제

Trident는 Kubernetes CRD를 사용하여 자체 상태를 저장 및 관리하고 Kubernetes 클러스터 etcd를 사용하여 메타데이터를 저장합니다.

단계

1. "Kubernetes: etcd 클러스터 백업"을 사용하여 Kubernetes 클러스터 etcd를 백업합니다.
2. 백업 아티팩트를 FlexVol 볼륨에 저장하세요.

> NetApp에서는 FlexVol이 있는 SVM을 SnapMirror 관계를 통해 다른 SVM으로 보호할 것을 권장합니다.

Trident 복구

Kubernetes CRD와 Kubernetes 클러스터 etcd 스냅샷을 사용하면 Trident를 복구할 수 있습니다.

단계

1. 타겟 SVM에서 Kubernetes etcd 데이터 파일과 인증서가 포함된 볼륨을 마스터 노드로 설정될 호스트에 마운트합니다.
2. Kubernetes 클러스터와 관련된 모든 필수 인증서를 `/etc/kubernetes/pki` 아래에 복사하고 etcd 멤버 파일을 `/var/lib/etcd` 아래에 복사하십시오.
3. "Kubernetes: etcd 클러스터 복원"을 사용하여 etcd 백업에서 Kubernetes 클러스터를 복원합니다.
4. `kubectl get crd`를 실행하여 모든 Trident 사용자 지정 리소스가 제대로 활성화되었는지 확인하고 Trident 객체를 검색하여 모든 데이터를 사용할 수 있는지 확인합니다.

SVM 복제 및 복구

Trident는 복제 관계를 구성할 수 없지만, 스토리지 관리자는 "ONTAP SnapMirror"를 사용하여 SVM을 복제할 수 있습니다.

재해가 발생할 경우 SnapMirror 타겟 SVM을 활성화하여 데이터 제공을 시작할 수 있습니다. 시스템 복구가 완료되면 운영 SVM으로 다시 전환할 수 있습니다.

이 작업 정보

SnapMirror SVM 복제 기능을 사용할 때 다음 사항을 고려하십시오.

- SVM-DR이 활성화된 각 SVM에 대해 별도의 백엔드를 생성해야 합니다.
- 필요한 경우에만 복제된 백엔드를 선택하도록 스토리지 클래스를 구성하여 SVM-DR을 지원하는 백엔드에 복제가 필요하지 않은 볼륨이 프로비저닝되지 않도록 하십시오.
- 애플리케이션 관리자는 복제와 관련된 추가 비용 및 복잡성을 이해하고 이 프로세스를 시작하기 전에 복구 계획을 신중하게 고려해야 합니다.

SVM 복제

"ONTAP: SnapMirror SVM 복제"를 사용하여 SVM 복제 관계를 생성할 수 있습니다.

SnapMirror를 사용하면 복제할 항목을 제어하는 옵션을 설정할 수 있습니다. Trident를 사용한 SVM 복구을 수행할 때 어떤 옵션을 선택했는지 알아야 합니다.

- "-identity-preserve true" 전체 SVM 구성을 복제합니다.
- "-discard-configs 네트워크" LIF 및 관련 네트워크 설정은 제외됩니다.
- "-identity-preserve false"볼륨과 보안 구성만 복제합니다.

Trident를 사용한 SVM 복구

Trident는 SVM 장애를 자동으로 감지하지 않습니다. 재해 발생 시 관리자는 수동으로 Trident 페일오버를 시작하여 새 SVM으로 전환할 수 있습니다.

단계

1. 예약 및 진행 중인 SnapMirror 전송을 취소하고, 복제 관계를 해제하고, 소스 SVM을 중지한 다음 SnapMirror 타겟 SVM을 활성화합니다.
2. SVM 복제를 구성할 때 `-identity-preserve false` 또는 `-discard-config network`를 지정한 경우 `Trident` 백엔드 정의 파일에서 `managementLIF` 및 `dataLIF`를 업데이트하십시오.
3. `storagePrefix`가 Trident 백엔드 정의 파일에 있는지 확인합니다. 이 매개변수는 변경할 수 없습니다. `storagePrefix`를 생략하면 백엔드 업데이트가 실패합니다.
4. 다음 명령어를 사용하여 필요한 모든 백엔드를 새 타겟 SVM 이름으로 업데이트하십시오.

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. `-identity-preserve false` 또는 `discard-config network`를 지정한 경우 모든 애플리케이션 Pod를 재시작해야 합니다.

> `-identity-preserve true`을 지정한 경우 타겟 SVM이 활성화되면 Trident에서 프로비저닝한 모든 볼륨이 데이터 제공을 시작합니다.

볼륨 복제 및 복구

Trident는 SnapMirror 복제 관계를 구성할 수 없지만 스토리지 관리자는 "ONTAP SnapMirror 복제 및 복구"를 사용하여 Trident에서 생성한 볼륨을 복제할 수 있습니다.

그런 다음 "tridentctl 볼륨 가져오기"을 사용하여 복구된 볼륨을 Trident로 가져올 수 있습니다.

> `ontap-nas-economy`, `ontap-san-economy` 또는 `ontap-flexgroup-economy` 드라이버에서는 가져오기가 지원되지 않습니다.

스냅샷 데이터 보호

다음 방법을 사용하여 데이터를 보호하고 복원할 수 있습니다.

- 영구 볼륨(PV)의 Kubernetes 볼륨 스냅샷을 생성하기 위한 외부 스냅샷 컨트롤러 및 CRD.

  "볼륨 스냅샷"

- ONTAP 스냅샷을 사용하여 볼륨의 전체 콘텐츠를 복원하거나 개별 파일 또는 LUN을 복구할 수 있습니다.

  "ONTAP 스냅샷"

Trident를 사용하여 상태 저장 애플리케이션의 페일오버 자동화

Trident의 강제 분리 기능은 Kubernetes 클러스터에서 비정상적인 노드로부터 볼륨을 자동으로 분리하여 데이터 손상을 방지하고 애플리케이션 가용성을 보장합니다. 이 기능은 노드가 응답하지 않거나 유지 보수를 위해 오프라인 상태가 되는 시나리오에서 특히 유용합니다.

강제 분리에 대한 세부 정보

강제 분리는 `ontap-san`, `ontap-san-economy`, `ontap-nas` 및 `ontap-nas-economy`에 대해서만 사용할 수 있습니다. 강제 분리를 활성화하기 전에 Kubernetes 클러스터에서 비정상 노드 종료(NGNS)를 활성화해야 합니다. NGNS는 Kubernetes 1.28 이상에서 기본적으로 활성화되어 있습니다. 자세한 내용은 "Kubernetes: 정상적이지 않은 노드 종료"을 참조하십시오.

> ⓘ `ontap-nas` 또는 `ontap-nas-economy` 드라이버를 사용할 때는 백엔드 구성에서 `autoExportPolicy` 매개 변수를 `true`로 설정해야 Trident가 관리형 내보내기 정책을 사용하여 테인트가 적용된 Kubernetes 노드의 액세스를 제한할 수 있습니다.

> ⓘ Trident는 Kubernetes NGNS에 의존하므로 모든 허용 불가능한 워크로드가 재배치될 때까지 비정상 노드에서 `out-of-service`테인트를 제거하지 마십시오. 테인트를 부적절하게 적용하거나 제거하면 백엔드 데이터 보호가 위험해질 수 있습니다.

Kubernetes 클러스터 관리자가 노드에 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 테인트를 적용하고 `enableForceDetach`가 `true`로 설정되면 Trident는 노드 상태를 확인하고 다음을 수행합니다.

1. 해당 노드에 마운트된 볼륨에 대한 백엔드 I/O 액세스를 중지합니다.
2. Trident 노드 객체를 `dirty`(새 게시에 안전하지 않음)로 표시합니다.

> ⓘ Trident 컨트롤러는 Trident 노드 Pod에서 노드가 재인증(`dirty`로 표시된 후)될 때까지 새로운 볼륨 게시 요청을 거부합니다. 마운트된 `PVC`를 사용하여 예약된 모든 워크로드(클러스터 노드가 정상 상태이고 준비된 후에도)는 `Trident`가 노드 `clean`(새로운 게시를 위한 안전한 상태)를 검증할 때까지 허용되지 않습니다.

노드 상태가 복구되고 오염이 제거되면 Trident는 다음과 같이 동작합니다.

1. 노드에서 더 이상 사용되지 않는 게시된 경로를 식별하고 정리합니다.

2. 노드가 `cleanable`상태(서비스 중단 오류 표시가 제거되고 노드가 `Ready`상태인 경우)이고 모든 오래된 게시 경로가 정리되면 Trident는 해당 노드를 `clean`로 다시 승인하고 해당 노드에 새 게시 볼륨을 허용합니다.

자동 페일오버에 대한 세부 정보

"노드 상태 점검(NHC) 운영자"와의 통합을 통해 강제 분리 프로세스를 자동화할 수 있습니다. 노드 장애가 발생하면 NHC는 Trident 노드 복구(TNR)를 트리거하고 장애가 발생한 노드를 정의하는 TridentNodeRemediation CR을 Trident 네임스페이스에 생성하여 자동으로 강제 분리를 수행합니다. TNR은 노드 장애 발생 시에만 생성되며, 노드가 다시 온라인 상태가 되거나 삭제되면 NHC에서 제거됩니다.

실패한 노드 **Pod** 제거 프로세스

자동 장애 조치는 장애가 발생한 노드에서 제거할 워크로드를 선택합니다. TNR이 생성되면 TNR 컨트롤러는 해당 노드를 비정상 상태로 표시하여 새로운 볼륨 게시를 방지하고, 강제 분리가 지원되는 Pod와 해당 Pod의 볼륨 연결을 제거하기 시작합니다.

강제 분리에서 지원하는 모든 볼륨/PVC는 자동 페일오버에서 지원됩니다.

- 자동 내보내기 정책을 사용하는 NAS 및 NAS-economy 볼륨(SMB는 아직 지원되지 않음)
- SAN 및 SAN-economy 볼륨.

강제 분리에 대한 세부 정보을 참조하십시오.

기본 동작:

- force-detach에서 지원하는 볼륨을 사용하는 Pod는 장애가 발생한 노드에서 제거됩니다. Kubernetes는 이러한 Pod를 정상 노드로 다시 스케줄링합니다.
- 강제 분리 기능을 지원하지 않는 볼륨(Trident 이외의 볼륨 포함)을 사용하는 Pod는 장애가 발생한 노드에서 제거되지 않습니다.
- 스테이트리스 파드(PVC 제외)는 파드 어노테이션 `trident.netapp.io/podRemediationPolicy: delete`이 설정되어 있지 않으면 장애가 발생한 노드에서 제거되지 않습니다.

Pod 제거 동작 재정의:

Pod 제거 동작은 Pod 어노테이션을 사용하여 사용자 지정할 수 있습니다 `trident.netapp.io/podRemediationPolicy[retain, delete]`. 이러한 어노테이션은 페일오버 발생 시 검사되고 사용됩니다. 페일오버 후 어노테이션이 사라지지 않도록 하려면 Kubernetes 배포/복제 세트 Pod 사양에 어노테이션을 적용하세요:

- `retain` - 자동 장애 조치 중에 Pod는 장애가 발생한 노드에서 제거되지 않습니다.
- `delete` - 자동 장애 조치 중에 Pod는 장애가 발생한 노드에서 제거됩니다.

이러한 주석은 모든 포드에 적용할 수 있습니다.

- 강제 분리를 지원하는 볼륨의 경우 장애가 발생한 노드에서만 I/O 작업이 차단됩니다.
- 강제 분리를 지원하지 않는 볼륨의 경우 데이터 손상 및 다중 연결 문제가 발생할 위험이 있습니다.

TridentNodeRemediation CR

TridentNodeRemediation(TNR) CR은 장애가 발생한 노드를 정의합니다. TNR의 이름은 장애가 발생한 노드의 이름입니다.

TNR 예:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR 상태: 다음 명령을 사용하여 TNR 상태를 확인합니다.
`kubectl get tnr <name> -n <trident-namespace>`

TNR은 다음 상태 중 하나에 있을 수 있습니다.

- 복구 중:
  - 해당 노드에 마운트된 force-detach에서 지원하는 볼륨에 대한 백엔드 I/O 액세스를 중지합니다.
  - Trident 노드 객체가 더티(새 게시에 안전하지 않음)로 표시됩니다.
  - 노드에서 Pod 및 볼륨 연결을 제거합니다
- *NodeRecoveryPending*:
  - 컨트롤러가 노드가 다시 온라인 상태가 되기를 기다리고 있습니다.
  - 노드가 온라인 상태가 되면 publish-enforcement를 통해 노드가 깨끗하고 새로운 볼륨 게시를 위한 준비가 완료되었는지 확인합니다.
- 노드가 K8s에서 삭제되면 TNR 컨트롤러는 TNR을 제거하고 조정 작업을 중지합니다.
- 성공:
  - 모든 복구 및 노드 복구 단계가 성공적으로 완료되었습니다. 노드가 정상이며 새 볼륨 게시 준비가 되었습니다.
- 실패:
  - 복구 불가능한 오류입니다. 오류 원인은 CR의 status.message 필드에 설정되어 있습니다.

자동 페일오버 활성화

사전 요구 사항:

- 자동 장애 조치를 활성화하기 전에 강제 분리가 활성화되어 있는지 확인하십시오. 자세한 내용은 강제 분리에 대한 세부 정보를 참조하십시오.
- Kubernetes 클러스터에 노드 상태 확인(NHC)을 설치합니다.
  - "operator-sdk 설치".
  - 클러스터에 Operator Lifecycle Manager(OLM)가 설치되어 있지 않은 경우 설치하십시오 `operator-sdk olm install`.

- Node Health check Operator를 설치합니다 `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`.

> 아래 [Integrating Custom Node Health Check Solutions] 섹션에 명시된 대로 노드 장애를 감지하는 다른 방법을 사용할 수도 있습니다.

자세한 내용은 "노드 상태 점검 오퍼레이터"를 참조하십시오.

단계

1. 클러스터의 워커 노드를 모니터링하기 위해 Trident 네임스페이스에 NodeHealthCheck(NHC) CR을 생성합니다. 예:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. `trident` 네임스페이스에 노드 상태 점검 CR을 적용합니다.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

위의 CR은 노드 조건 Ready: false 및 Unknown에 대해 K8s 워커 노드를 감시하도록 구성되어 있습니다. 노드가 Ready: false 또는 Ready: Unknown 상태가 되면 Automated-Failover가 트리거됩니다.

CR의 `unhealthyConditions`는 0초 유예 기간을 사용합니다. 이로 인해 K8s가 노드에서 하트비트를 수신하지 못한 후 노드 조건을 Ready: false로 설정하면 즉시 자동 페일오버가 트리거됩니다. K8s는 기본적으로 마지막 하트비트 후

40초를 대기한 후 Ready: false로 설정합니다. 이 유예 기간은 K8s 배포 옵션에서 사용자 지정할 수 있습니다.

추가 구성 옵션은 "Node-Healthcheck-Operator 문서"을 참조하십시오.

추가 설정 정보

Trident가 강제 분리 기능을 활성화하여 설치되면 NHC와의 통합을 용이하게 하기 위해 Trident 네임스페이스에 두 개의 추가 리소스가 자동으로 생성됩니다: TridentNodeRemediationTemplate(TNRT) 및 ClusterRole.

TridentNodeRemediationTemplate(TNRT):

TNRT는 NHC 컨트롤러의 템플릿 역할을 하며, TNRT를 사용하여 필요에 따라 TNR 리소스를 생성합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole:

force-detach가 활성화되면 설치 중에 클러스터 역할이 추가됩니다. 이를 통해 NHC는 Trident 네임스페이스의 TNR에 대한 권한을 갖게 됩니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s 클러스터 업그레이드 및 유지 관리

장애 조치를 방지하려면 노드가 다운되거나 재부팅될 것으로 예상되는 K8s 유지 관리 또는 업그레이드 중에 자동 장애 조치를 일시 중지하십시오. 위에서 설명한 NHC CR을 일시 중지하려면 해당 CR에 패치를 적용하면 됩니다.

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

이렇게 하면 자동 장애 조치가 일시 중지됩니다. 자동 장애 조치를 다시 활성화하려면 유지 관리가 완료된 후 사양에서 pauseRequests를 제거하십시오.

제한 사항

- I/O 작업은 force-detach에서 지원하는 볼륨에 대해 장애가 발생한 노드에서만 차단됩니다. force-detach에서 지원하는 볼륨/PVC를 사용하는 Pod만 자동으로 제거됩니다.
- 자동 페일오버 및 강제 분리 기능은 trident-controller pod 내부에서 실행됩니다. trident-controller를 호스팅하는 노드에 장애가 발생하면 K8s가 pod를 정상 노드로 이동할 때까지 자동 페일오버가 지연됩니다.

사용자 정의 노드 상태 확인 솔루션 통합

Node Healthcheck Operator를 대체 노드 장애 감지 도구로 교체하여 자동 장애 조치를 트리거할 수 있습니다. 자동 장애 조치 메커니즘과의 호환성을 보장하려면 사용자 지정 솔루션은 다음을 수행해야 합니다.

- 노드 장애가 감지되면 장애가 발생한 노드의 이름을 TNR CR 이름으로 사용하여 TNR을 생성합니다.
- 노드가 복구되고 TNR이 Succeeded 상태일 때 TNR을 삭제합니다.

보안

보안

여기에 나열된 권장 사항을 사용하여 Trident 설치의 보안을 확보하십시오.

Trident를 자체 네임스페이스에서 실행합니다

안정적인 스토리지를 보장하고 잠재적인 악의적 활동을 차단하려면 애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션이 Trident 객체 정의 또는 Pod에 액세스하는 것을 방지하는 것이 중요합니다.

다른 애플리케이션 및 사용자와 Trident를 분리하려면 항상 Trident를 자체 Kubernetes 네임스페이스에 설치하십시오(`trident`. Trident를 자체 네임스페이스에 배치하면 Kubernetes 관리 담당자만 Trident Pod 및 네임스페이스 CRD 객체에 저장된 아티팩트(해당하는 경우 백엔드 및 CHAP 암호)에 액세스할 수 있습니다. 관리자만 Trident 네임스페이스에 액세스하여 `tridentctl` 애플리케이션에 액세스할 수 있도록 해야 합니다.

ONTAP SAN 백엔드에서 **CHAP** 인증을 사용하십시오

Trident는 ONTAP SAN 워크로드에 대해 CHAP 기반 인증을 지원합니다(`ontap-san` 및 `ontap-san-economy` 드라이버 사용). NetApp은 호스트와 스토리지 백엔드 간의 인증을 위해 Trident와 함께 양방향 CHAP를 사용하는 것을 권장합니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Trident는 `tridentctl`를 통해 양방향 CHAP를 설정하고 CHAP 사용자 이름과 암호를 관리할 수 있습니다. Trident가 ONTAP 백엔드에서 CHAP를 구성하는 방법을 이해하려면"ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 하십시오"를 참조하십시오.

NetApp HCI 및 **SolidFire** 백엔드에서 **CHAP** 인증을 사용하십시오

NetApp에서는 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하기 위해 양방향 CHAP 배포를 권장합니다. Trident는 테넌트당 두 개의 CHAP 암호를 포함하는 secret 객체를 사용합니다. Trident가 설치되면 CHAP secret을 관리하고 해당 PV에 대한 `tridentvolume` CR 객체에 저장합니다. PV를 생성하면 Trident는 CHAP secret을 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

Trident에서 생성한 볼륨은 어떤 볼륨 액세스 그룹과도 연결되지 않습니다.

NVE 및 **NAE**와 함께 **Trident** 사용

NetApp ONTAP는 디스크가 도난당하거나 반환되거나 용도가 변경될 경우 중요한 데이터를 보호하기 위해 저장 데이터 암호화를 제공합니다. 자세한 내용은 "NetApp Volume Encryption 구성 개요"를 참조하십시오.

- 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다.
  - NVE encryption 플래그를 `""`로 설정하여 NAE 지원 볼륨을 생성할 수 있습니다.
- 백엔드에서 NAE가 활성화되지 않은 경우 백엔드 구성에서 NVE 암호화 플래그가 `false`(기본값)로 설정되지 않는 한 Trident에서 프로비저닝된 모든 볼륨은 NVE가 활성화됩니다.

NAE가 활성화된 백엔드에서 Trident로 생성된 볼륨은 NVE 또는 NAE로 암호화되어야 합니다.

- Trident 백엔드 구성에서 NVE 암호화 플래그를 `true`로 설정하여 NAE 암호화를 재정의하고 볼륨별로 특정 암호화 키를 사용할 수 있습니다.
- NAE가 활성화된 백엔드에서 NVE 암호화 플래그를 `false`로 설정하면 NAE가 활성화된 볼륨이 생성됩니다. NVE 암호화 플래그를 `false`로 설정하여 NAE 암호화를 비활성화할 수는 없습니다.

- Trident에서 NVE 암호화 플래그를 명시적으로 `true`로 설정하여 수동으로 NVE 볼륨을 생성할 수 있습니다.

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하십시오.

- "ONTAP SAN 구성 옵션"
- "ONTAP NAS 구성 옵션"

Linux Unified Key Setup(LUKS)

Trident에서 Linux Unified Key Setup(LUKS)을 활성화하여 ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨을 암호화할 수 있습니다. Trident는 LUKS로 암호화된 볼륨에 대해 암호 순환 및 볼륨 확장을 지원합니다.

Trident에서 LUKS로 암호화된 볼륨은 "NIST"에서 권장하는 대로 aes-xts-plain64 암호 및 모드를 사용합니다.

LUKS 암호화는 ASA r2 시스템에서 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 "ASA r2 스토리지 시스템에 대해 알아보세요"를 참조하십시오.

시작하기 전에

- 워커 노드에는 cryptsetup 2.1 이상(단, 3.0 미만)이 설치되어 있어야 합니다. 자세한 내용은 "Gitlab: cryptsetup"를 참조하십시오.
- 성능 향상을 위해 NetApp에서는 워커 노드가 AES-NI(Advanced Encryption Standard New Instructions)를 지원하도록 권장합니다. AES-NI 지원 여부를 확인하려면 다음 명령을 실행하십시오.

```
grep "aes" /proc/cpuinfo
```

아무것도 반환되지 않으면 프로세서가 AES-NI를 지원하지 않는 것입니다. AES-NI에 대한 자세한 내용은 "Intel: Advanced Encryption Standard Instructions(AES-NI)"를 참조하십시오.

LUKS 암호화 활성화

ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨의 경우 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 "ONTAP SAN 구성 옵션"을 참조하십시오.

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. `parameters.selector`를 사용하여 LUKS 암호화를 사용하는 스토리지 풀을 정의합니다. 예를 들면 다음과 같습니다.

```
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 암호를 포함하는 비밀 키를 생성합니다. 예를 들면 다음과 같습니다.

```
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

LUKS로 암호화된 볼륨은 ONTAP 중복 제거 및 압축을 활용할 수 없습니다.

LUKS 볼륨 가져오기를 위한 백엔드 구성

LUKS 볼륨을 가져오려면 백엔드에서 `luksEncryption`을 (`true`로 설정해야 합니다. `luksEncryption` 옵션은 다음 예와 같이 볼륨이 LUKS 규격을 준수하는지(`true` 또는 LUKS 규격을 준수하지 않는지(`false`를 Trident에 알려줍니다.

```
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 annotation `trident.netapp.io/luksEncryption`을 `true`로 설정하고 이 예제와 같이 PVC에 LUKS 지원 스토리지 클래스를 포함하십시오.

```
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 교체

LUKS 암호를 교체하고 교체를 확인할 수 있습니다.

> 볼륨, 스냅샷 또는 시크릿에서 더 이상 참조되지 않는다는 것을 확인하기 전까지는 암호를 잊지 마십시오. 참조된 암호를 분실하면 볼륨을 마운트할 수 없으며 데이터는 암호화된 상태로 유지되어 액세스할 수 없게 됩니다.

이 작업 정보

LUKS 암호 순환은 새 LUKS 암호가 지정된 후 해당 볼륨을 마운트하는 Pod가 생성될 때 발생합니다. 새 Pod가 생성될 때 Trident는 볼륨의 LUKS 암호를 시크릿에 있는 활성 암호와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 순환이 발생합니다.
- 볼륨의 암호가 비밀 키에 있는 활성 암호와 일치하면 `previous-luks-passphrase` 매개변수는 무시됩니다.

단계

1. `node-publish-secret-name` 및 `node-publish-secret-namespace` StorageClass 매개변수를 추가합니다. 예를 들면 다음과 같습니다.

```
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. 볼륨 또는 스냅샷에 있는 기존 암호를 확인합니다.

볼륨

```
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 암호를 업데이트하여 새 암호와 이전 암호를 지정하십시오. `previous-luke-passphrase-name`와 `previous-luks-passphrase`가 이전 암호와 일치하는지 확인하십시오.

```
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 마운트하는 새 Pod를 생성합니다. 이는 로테이션을 시작하는 데 필요합니다.
5. 암호가 교체되었는지 확인하십시오.

볼륨

```
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

볼륨 및 스냅샷에 새 암호만 반환되면 암호가 교체된 것입니다.

> 예를 들어 두 개의 암호문이 반환되면 `luksPassphraseNames: ["B", "A"]` 로테이션이 완료되지 않은 것입니다. 로테이션을 완료하기 위해 새 Pod를 트리거할 수 있습니다.

볼륨 확장 활성화

LUKS로 암호화된 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. `CSINodeExpandSecret` 기능 게이트를 활성화합니다(베타 1.25+). 자세한 내용은 "Kubernetes 1.25: 노드 기반 CSI 볼륨 확장을 위한 시크릿 사용"를 참조하십시오.
2. `node-expand-secret-name` 및 `node-expand-secret-namespace` StorageClass 매개변수를 추가합니다. 예를 들면 다음과 같습니다.

```
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 kubelet이 적절한 자격 증명을 드라이버에 전달합니다.

Kerberos 전송 중 암호화

Kerberos 전송 중 암호화를 사용하면 관리형 클러스터와 스토리지 백엔드 간의 트래픽에 대한 암호화를 활성화하여 데이터 액세스 보안을 개선할 수 있습니다.

Trident는 스토리지 백엔드로 ONTAP에 대한 Kerberos 암호화를 지원합니다.

- 온프레미스 **ONTAP** - Trident는 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, 클론 복제, 읽기 전용 클론 복제 및 가져오기를 수행할 수 있습니다.

온프레미스 **ONTAP** 볼륨을 사용하여 전송 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 `ontap-nas` 스토리지 드라이버를 통해서만 지원됩니다.

시작하기 전에

- `tridentctl` 유틸리티에 액세스할 수 있는지 확인하십시오.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있는지 확인하십시오.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있는지 확인하십시오.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하도록 ONTAP 스토리지 VM을 준비했는지 확인하십시오. 지침은 "dataLIF에서 Kerberos를 활성화합니다"을 참조하십시오.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하십시오. "NetApp NFSv4 향상 기능 및 모범 사례 가이드"의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오.

ONTAP 내보내기 정책을 추가 또는 수정합니다

기존 ONTAP 내보내기 정책에 규칙을 추가하거나 ONTAP 스토리지 VM 루트 볼륨과 업스트림 Kubernetes 클러스터와 공유되는 모든 ONTAP 볼륨에 대해 Kerberos 암호화를 지원하는 새 내보내기 정책을 생성해야 합니다. 추가하는 내보내기 정책 규칙 또는 새로 생성하는 내보내기 정책은 다음 액세스 프로토콜 및 액세스 권한을 지원해야 합니다.

액세스 프로토콜

NFS, NFSv3 및 NFSv4 액세스 프로토콜을 사용하여 엑스포트 정책을 구성합니다.

액세스 세부 정보

볼륨에 대한 요구 사항에 따라 세 가지 버전의 Kerberos 암호화 중 하나를 구성할 수 있습니다.

- **Kerberos 5** - (인증 및 암호화)
- **Kerberos 5i** - (ID 보호 기능을 갖춘 인증 및 암호화)

- **Kerberos 5p** - (신원 및 개인정보 보호를 통한 인증 및 암호화)

적절한 액세스 권한으로 ONTAP 엑스포트 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i와 Kerberos 5p 암호화를 혼합하여 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용하십시오.

유형	읽기 전용 액세스	읽기/쓰기 액세스	슈퍼유저 액세스
UNIX	활성화됨	활성화됨	활성화됨
Kerberos 5i	활성화됨	활성화됨	활성화됨
Kerberos 5p	활성화됨	활성화됨	활성화됨

ONTAP 내보내기 정책 및 내보내기 정책 규칙을 생성하는 방법은 다음 문서를 참조하십시오.

- "엑스포트 정책을 생성합니다"
- "엑스포트 정책에 규칙 추가"

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Trident 스토리지 백엔드 구성을 생성할 수 있습니다.

이 작업 정보

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 생성할 때 `spec.nfsMountOptions` 매개변수를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다.

- `spec.nfsMountOptions: sec=krb5` (인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5i` (ID 보호 기능이 있는 인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5p` (ID 및 개인 정보 보호를 통한 인증 및 암호화)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에 두 개 이상의 Kerberos 암호화 수준을 지정하면 첫 번째 옵션만 사용됩니다.

단계

1. 관리형 클러스터에서 다음 예제를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 <> 안의 값을 사용자 환경의 정보로 바꾸십시오.

```
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 생성한 구성 파일을 사용하여 백엔드를 생성하십시오.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 식별하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하는 스토리지 클래스를 생성할 수 있습니다.

이 작업 정보

스토리지 클래스 객체를 생성할 때 `mountOptions` 매개변수를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 있습니다.

- `mountOptions: sec=krb5` (인증 및 암호화)
- `mountOptions: sec=krb5i` (ID 보호를 통한 인증 및 암호화)
- `mountOptions: sec=krb5p` (ID 및 개인 정보 보호를 통한 인증 및 암호화)

Kerberos 수준을 하나만 지정하십시오. 매개변수 목록에 두 개 이상의 Kerberos 암호화 수준을 지정하면 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 객체에서 지정한 수준과 다른 경우 스토리지 클래스 객체가 우선합니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 객체를 생성합니다.

```
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. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 스토리지 클래스가 생성되었는지 확인하십시오.

```
kubectl get sc ontap-nas-sc
```

다음과 유사한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후에는 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 "볼륨 프로비저닝"를 참조하십시오.

Azure NetApp Files 볼륨에서 전송 중 Kerberos 암호화 구성

관리형 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 간의 스토리지 트래픽에 대해 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리형 Red Hat OpenShift 클러스터에서 Trident가 활성화되어 있는지 확인하십시오.
- `tridentctl` 유틸리티에 액세스할 수 있는지 확인하십시오.
- 요구 사항을 확인하고 "Azure NetApp Files 설명서"의 지침에 따라 Kerberos 암호화를 위한 Azure NetApp Files 스토리지 백엔드를 준비했는지 확인하십시오.
- Kerberos 암호화와 함께 사용하는 모든 NFSv4 볼륨이 올바르게 구성되었는지 확인하십시오. "NetApp NFSv4 향상 기능 및 모범 사례 가이드"의 NetApp NFSv4 도메인 구성 섹션(13페이지)을 참조하십시오.

스토리지 백엔드를 생성합니다

Kerberos 암호화 기능을 포함하는 Azure NetApp Files 스토리지 백엔드 구성을 만들 수 있습니다.

이 작업 정보

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 생성할 때 다음 두 가지 수준 중 하나에 적용되도록 정의할 수 있습니다.

- `spec.kerberos` 필드를 사용하는 스토리지 백엔드 레벨
- `spec.storage.kerberos` 필드를 사용한 가상 풀 레벨

가상 풀 수준에서 구성을 정의할 때 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

어느 수준에서든 세 가지 버전의 Kerberos 암호화 중 하나를 지정할 수 있습니다.

- `kerberos: sec=krb5` (인증 및 암호화)
- `kerberos: sec=krb5i` (ID 보호를 통한 인증 및 암호화)
- `kerberos: sec=krb5p` (ID 및 개인 정보 보호를 통한 인증 및 암호화)

단계

1. 관리형 클러스터에서 스토리지 백엔드를 정의해야 하는 위치(스토리지 백엔드 수준 또는 가상 풀 수준)에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 <> 안의 값은 사용자 환경에 맞는 정보로 바꾸십시오.

스토리지 백엔드 수준 예

```
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 수준 예

```
---
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 생성한 구성 파일을 사용하여 백엔드를 생성하십시오.

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

백엔드 생성에 실패하면 백엔드 구성에 문제가 있는 것입니다. 다음 명령을 실행하여 로그를 확인하고 원인을 파악할 수 있습니다.

```
tridentctl logs
```

구성 파일의 문제를 식별하고 수정한 후 create 명령을 다시 실행할 수 있습니다.

스토리지 클래스를 생성합니다

Kerberos 암호화를 사용하여 볼륨을 프로비저닝하는 스토리지 클래스를 생성할 수 있습니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 객체를 생성합니다.

```
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. 스토리지 클래스를 생성합니다.

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. 스토리지 클래스가 생성되었는지 확인하십시오.

```
kubectl get sc -sc-nfs
```

다음과 유사한 출력이 표시됩니다.

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

볼륨 프로비저닝

스토리지 백엔드와 스토리지 클래스를 생성한 후에는 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 "볼륨 프로비저닝"를 참조하십시오.

Trident Protect로 애플리케이션을 보호하세요

Trident Protect에 대해 알아보십시오

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너를 기반으로 하는 스테이트풀 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다. Trident Protect는 퍼블릭 클라우드와 온프레미스 환경 전반에 걸쳐 컨테이너화된 워크로드의 관리, 보호 및 이동을 간소화합니다. 또한 API 및 CLI를 통해 자동화 기능을 제공합니다.

사용자 지정 리소스(CR)를 생성하거나 Trident Protect CLI를 사용하여 Trident Protect로 애플리케이션을 보호할 수 있습니다.

다음 단계

Trident Protect를 설치하기 전에 요구 사항에 대해 알아볼 수 있습니다.

- "Trident Protect 요구 사항"

Trident Protect를 설치합니다

Trident Protect 요구 사항

먼저 운영 환경, 애플리케이션 클러스터, 애플리케이션 및 라이선스의 준비 상태를 확인하십시오. Trident Protect를 배포하고 운영하기 위한 요구 사항을 환경이 충족하는지 확인하십시오.

Trident Protect Kubernetes 클러스터 호환성

Trident Protect는 다음을 포함한 다양한 완전 관리형 및 자체 관리형 Kubernetes 제품과 호환됩니다.

- Amazon Elastic Kubernetes Service(EKS)
- Google Kubernetes Engine(GKE)
- Microsoft Azure Kubernetes Service(AKS)
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- 업스트림 Kubernetes

- Trident Protect 백업은 Linux 컴퓨팅 노드에서만 지원됩니다. Windows 컴퓨팅 노드에서는 백업 작업이 지원되지 않습니다.
- Trident Protect를 설치할 클러스터에 실행 중인 스냅샷 컨트롤러와 관련 CRD가 구성되어 있는지 확인하십시오. 스냅샷 컨트롤러를 설치하려면 "이 지침"를 참조하십시오.
- 하나 이상의 VolumeSnapshotClass가 존재하는지 확인하십시오. 자세한 내용은 "VolumeSnapshotClass"을 참조하십시오.

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 확인하십시오.

- 클러스터에 연결된 NetApp 스토리지가 Trident 24.02 이상 버전(Trident 24.10 권장)을 사용하고 있는지 확인하십시오.
- NetApp ONTAP 스토리지 백엔드가 있는지 확인하십시오.
- 백업 저장을 위한 오브젝트 스토리지 버킷을 구성했는지 확인하십시오.
- 애플리케이션 또는 애플리케이션 데이터 관리 작업에 사용할 애플리케이션 네임스페이스를 생성하십시오. Trident Protect는 이러한 네임스페이스를 자동으로 생성하지 않으므로 사용자 지정 리소스에 존재하지 않는 네임스페이스를 지정하면 작업이 실패합니다.

nas-economy 볼륨에 대한 요구 사항

Trident Protect는 nas-economy 볼륨에 대한 백업 및 복원 작업을 지원합니다. 스냅샷, 클론 및 SnapMirror 복제는 현재 nas-economy 볼륨에 대해 지원되지 않습니다. Trident Protect에서 사용할 각 nas-economy 볼륨에 대해 스냅샷 디렉터리를 활성화해야 합니다.

일부 애플리케이션은 스냅샷 디렉터리를 사용하는 볼륨과 호환되지 않습니다. 이러한 애플리케이션의 경우 ONTAP 스토리지 시스템에서 다음 명령을 실행하여 스냅샷 디렉터리를 숨겨야 합니다.

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

각 nas-economy 볼륨에 대해 다음 명령을 실행하여 스냅샷 디렉터리를 활성화할 수 있습니다. `<volume-UUID>`을(를) 변경하려는 볼륨의 UUID로 바꾸십시오.

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Trident 백엔드 구성 옵션 `snapshotDir`을 `true`로 설정하여 새 볼륨에 대해 스냅샷 디렉터리를 기본적으로 활성화할 수 있습니다. 기존 볼륨에는 영향을 미치지 않습니다.

KubeVirt VM으로 데이터 보호

Trident Protect는 데이터 보호 작업 중 KubeVirt 가상 머신에 대한 파일 시스템 동결 및 동결 해제 기능을 제공하여 데이터 일관성을 보장합니다. VM 동결 작업에 대한 구성 방법과 기본 동작은 Trident Protect 버전에 따라 다르며, 최신 릴리스에서는 Helm 차트 매개변수를 통해 간소화된 구성을 제공합니다.

복원 작업 중에는 가상 머신(VM)에 대해 생성된 `VirtualMachineSnapshots`는 복원되지 않습니다.

Trident Protect 25.10 이상

Trident Protect는 데이터 보호 작업 중에 일관성을 보장하기 위해 KubeVirt 파일 시스템을 자동으로 동결 및 해제합니다. Trident Protect 25.10부터는 Helm 차트 설치 시 `vm.freeze` 매개변수를 사용하여 이 동작을 비활성화할 수 있습니다. 이 매개변수는 기본적으로 활성화되어 있습니다.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 ~ 25.06

Trident Protect 24.10.1부터 Trident Protect는 데이터 보호 작업 중에 KubeVirt 파일 시스템을 자동으로 동결 및 해제합니다. 필요에 따라 다음 명령을 사용하여 이 자동 동작을 비활성화할 수 있습니다:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 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 수행하기 전에 파일 시스템의 동결/해제 기능을 수동으로 활성화해야 합니다. 이렇게 하면 파일 시스템이 일관된 상태를 유지하게 됩니다.

Trident Protect 24.10을 구성하여 "가상화 구성"을(를) 사용한 다음 다음 명령을 사용하여 데이터 보호 작업 중 VM 파일 시스템의 동결 및 해제를 관리할 수 있습니다.

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

SnapMirror 복제 요구 사항

NetApp SnapMirror 복제는 다음 ONTAP 솔루션에서 Trident Protect와 함께 사용할 수 있습니다.

- 온프레미스 NetApp FAS, AFF 및 ASA 시스템. SnapMirror Trident Protect를 사용한 복제는 현재 ASA r2 시스템에서 지원되지 않습니다.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP 클러스터 요구 사항 SnapMirror 복제

SnapMirror 복제를 사용하려는 경우 ONTAP 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- **NetApp Trident**: NetApp Trident는 ONTAP를 백엔드로 사용하는 소스 및 타겟 Kubernetes 클러스터 모두에 존재해야 합니다. Trident Protect는 다음 드라이버를 기반으로 하는 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 타겟 ONTAP 클러스터 모두에서 활성화되어야 합니다. 자세한 내용은 "SnapMirror ONTAP 라이센싱 개요"를 참조하십시오.

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

  > SnapMirror 비동기 보호 기능만 지원됩니다.

SnapMirror 복제를 위한 피어링 고려 사항

스토리지 백엔드 피어링을 사용하려는 경우 환경이 다음 요구 사항을 충족하는지 확인하십시오.

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

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

- **NetApp Trident** 및 **SVM**: 피어링된 원격 SVM은 타겟 클러스터의 NetApp Trident에서 사용할 수 있어야 합니다.
- 관리형 백엔드: 복제 관계를 생성하려면 Trident Protect에 ONTAP 스토리지 백엔드를 추가하고 관리해야 합니다.

SnapMirror 복제를 위한 Trident/ONTAP 구성

Trident Protect를 사용하려면 소스 클러스터와 타겟 클러스터 모두에 대해 복제를 지원하는 스토리지 백엔드를 하나 이상 구성해야 합니다. 소스 클러스터와 타겟 클러스터가 동일한 경우, 최상의 복원력을 위해 타겟 애플리케이션은 소스 애플리케이션과 다른 스토리지 백엔드를 사용해야 합니다.

SnapMirror 복제를 위한 Kubernetes 클러스터 요구 사항

Kubernetes 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- **AppVault** 접근성: 소스 및 타겟 클러스터 모두 애플리케이션 객체 복제를 위해 AppVault에서 읽고 쓰기 위한 네트워크 액세스가 필요합니다.
- 네트워크 연결: 방화벽 규칙, 버킷 권한 및 IP 허용 목록을 구성하여 두 클러스터와 AppVault 간의 WAN 통신을 활성화합니다.

많은 기업 환경에서는 WAN 연결 전반에 걸쳐 엄격한 방화벽 정책을 시행합니다. 복제를 구성하기 전에 인프라 팀과 함께 이러한 네트워크 요구 사항을 확인하십시오.

Trident Protect를 설치하고 구성합니다**.**

사용 환경이 Trident Protect의 요구 사항을 충족하는 경우 다음 단계에 따라 클러스터에 Trident Protect를 설치할 수 있습니다. Trident Protect는 NetApp에서 얻거나 자체 프라이빗 레지스트리에서 설치할 수 있습니다. 클러스터가 인터넷에 액세스할 수 없는 경우 프라이빗 레지스트리에서 설치하는 것이 유용합니다.

Trident Protect를 설치합니다

NetApp에서 **Trident Protect** 설치

단계

1. Trident Helm 리포지토리 추가:

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

2. Helm을 사용하여 Trident Protect를 설치합니다. `<name-of-cluster>`를 클러스터 이름으로 바꾸세요. 이 이름은 클러스터에 할당되어 클러스터의 백업 데이터 및 스냅샷을 식별하는 데 사용됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. 선택적으로 디버그 로깅을 활성화하려면(문제 해결 권장) 다음을 사용하십시오.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

디버그 로깅은 NetApp 지원팀이 로그 레벨 변경이나 문제 재현 없이 문제를 해결하는 데 도움이 됩니다.

개인 레지스트리에서 **Trident Protect**를 설치하세요

Kubernetes 클러스터가 인터넷에 액세스할 수 없는 경우 개인 이미지 레지스트리에서 Trident Protect를 설치할 수 있습니다. 다음 예에서 괄호 안의 값은 사용자 환경에 맞는 정보로 바꾸십시오.

단계

1. 다음 이미지를 로컬 시스템으로 가져오고 태그를 업데이트한 다음 프라이빗 레지스트리로 푸시합니다.

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

예를 들면 다음과 같습니다.

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Helm 차트를 얻으려면 먼저 인터넷에 연결된 컴퓨터에서 `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`를 사용하여 Helm 차트를 다운로드한 다음, 생성된 `trident-protect-100.2510.0.tgz` 파일을 오프라인 환경으로 복사하고 마지막 단계에서 저장소 참조 대신 `helm install trident-protect ./trident-protect-100.2510.0.tgz`를 사용하여 설치하십시오.

2. Trident Protect 시스템 네임스페이스를 생성합니다.

```
kubectl create ns trident-protect
```

3. 레지스트리에 로그인합니다.

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 프라이빗 레지스트리 인증에 사용할 풀 시크릿을 생성합니다.

```
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 리포지토리 추가:

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

6. `protectValues.yaml`라는 이름의 파일을 생성하세요. 해당 파일에 다음 Trident Protect 설정이 포함되어 있는지 확인하세요.

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

imageRegistry 및 imagePullSecrets 값은 resourcebackup 및 resourcerestore`를 포함한 모든 구성 요소 이미지에 적용됩니다. 레지스트리 내의 특정 저장소 경로(예: `example.com:443/my-repo)에 이미지를 푸시하는 경우 레지스트리 필드에 전체 경로를 포함하십시오. 이렇게 하면 모든 이미지가 `<private-registry-url>/<image-name>:<tag>`에서 가져와집니다.

7. Helm을 사용하여 Trident Protect를 설치합니다. `<name_of_cluster>`를 클러스터 이름으로 바꾸세요. 이 이름은 클러스터에 할당되어 클러스터의 백업 데이터 및 스냅샷을 식별하는 데 사용됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. 선택적으로 디버그 로깅을 활성화하려면(문제 해결 권장) 다음을 사용하십시오.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

디버그 로깅은 NetApp 지원팀이 로그 레벨 변경이나 문제 재현 없이 문제를 해결하는 데 도움이 됩니다.

AutoSupport 설정 및 네임스페이스 필터링을 포함한 추가 Helm 차트 구성 옵션은 "Trident Protect 설치 사용자 지정"를 참조하십시오.

Trident Protect CLI 플러그인을 설치합니다

Trident tridentctl 유틸리티의 확장 기능인 Trident Protect 명령줄 플러그인을 사용하여 Trident Protect 사용자 정의 리소스(CR)를 생성하고 상호 작용할 수 있습니다.

Trident Protect CLI 플러그인을 설치합니다

명령줄 유틸리티를 사용하기 전에 클러스터에 액세스하는 데 사용하는 시스템에 유틸리티를 설치해야 합니다. 시스템에서 x64 또는 ARM CPU를 사용하는지 여부에 따라 다음 단계를 수행합니다.

Linux AMD64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드합니다.

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

Linux ARM64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드합니다.

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

Mac AMD64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드합니다.

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

Mac ARM64 CPU용 플러그인 다운로드

단계

1. Trident Protect CLI 플러그인을 다운로드합니다.

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. 플러그인 바이너리에 대한 실행 권한을 활성화합니다.

```
chmod +x tridentctl-protect
```

2. PATH 변수에 정의된 위치에 플러그인 바이너리를 복사합니다. 예를 들어 `/usr/bin` 또는 `/usr/local/bin` (상승된 권한이 필요할 수 있음):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 선택적으로 플러그인 바이너리를 홈 디렉토리의 위치에 복사할 수 있습니다. 이 경우 해당 위치가 PATH 변수의 일부인지 확인하는 것이 좋습니다:

```
cp ./tridentctl-protect ~/bin/
```

PATH 변수의 위치에 플러그인을 복사하면 어느 위치에서나 `tridentctl-protect` 또는 `tridentctl protect`을 입력하여 플러그인을 사용할 수 있습니다.

Trident CLI 플러그인 도움말 보기

내장된 플러그인 도움말 기능을 사용하여 플러그인의 기능에 대한 자세한 도움말을 얻을 수 있습니다.

단계

1. 도움말 기능을 사용하여 사용 지침을 확인합니다.

```
tridentctl-protect help
```

명령 자동 완성 활성화

Trident Protect CLI 플러그인을 설치한 후 특정 명령에 대해 자동 완성을 활성화할 수 있습니다.

Bash 셸에 대한 자동 완성 활성화

단계

1. 완료 스크립트를 생성합니다.

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. 스크립트를 포함할 새 디렉토리를 홈 디렉토리에 만듭니다.

```
mkdir -p ~/.bash/completions
```

3. 다운로드한 스크립트를 `~/.bash/completions` 디렉터리로 이동합니다.

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 홈 디렉토리의 `~/.bashrc` 파일에 다음 줄을 추가합니다.

```
source ~/.bash/completions/tridentctl-completion.bash
```

Z 셸에 대한 자동 완성 활성화

단계

1. 완료 스크립트를 생성합니다.

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. 스크립트를 포함할 새 디렉토리를 홈 디렉토리에 만듭니다.

```
mkdir -p ~/.zsh/completions
```

3. 다운로드한 스크립트를 `~/.zsh/completions` 디렉토리로 이동합니다.

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 홈 디렉토리의 `~/.zprofile` 파일에 다음 줄을 추가합니다.

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

결과

다음 셸 로그인 시 tridentctl-protect 플러그인으로 명령 자동 완성을 사용할 수 있습니다.

Trident Protect 설치 사용자 지정

사용자 환경의 특정 요구 사항을 충족하도록 Trident Protect의 기본 구성을 사용자 지정할 수 있습니다.

Trident Protect 컨테이너 리소스 제한 지정

Trident Protect를 설치한 후 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 리소스 제한을 지정할 수 있습니다. 리소스 제한을 설정하면 Trident Protect 작업에서 소비하는 클러스터 리소스의 양을 제어할 수 있습니다.

단계

1. `resourceLimits.yaml`이라는 파일을 만듭니다.
2. 환경의 요구 사항에 따라 Trident Protect 컨테이너에 대한 리소스 제한 옵션으로 파일을 채웁니다.

   다음 예제 구성 파일에는 사용 가능한 설정이 표시되어 있으며 각 리소스 제한의 기본값이 포함되어 있습니다.

```
---
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. `resourceLimits.yaml` 파일의 값을 적용합니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

보안 컨텍스트 제약 조건 사용자 지정

구성 파일을 사용하여 Trident Protect 설치 후 Trident Protect 컨테이너에 대한 OpenShift 보안 컨텍스트 제약 조건(SCC)을 수정할 수 있습니다. 이러한 제약 조건은 Red Hat OpenShift 클러스터의 Pod에 대한 보안 제한을 정의합니다.

단계

1. `sccconfig.yaml`이라는 파일을 만듭니다.
2. 파일에 SCC 옵션을 추가하고 환경 요구 사항에 따라 매개 변수를 수정합니다.

   다음 예는 SCC 옵션의 매개변수 기본값을 보여줍니다.

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

   이 표에서는 SCC 옵션의 매개 변수에 대해 설명합니다.

매개변수	설명	기본값
생성	SCC 리소스를 생성할 수 있는지 여부를 결정합니다. `scc.create`이 `true`로 설정되어 있고 Helm 설치 프로세스가 OpenShift 환경을 식별하는 경우에만 SCC 리소스가 생성됩니다. OpenShift에서 작동하지 않거나 `scc.create`이 `false`로 설정되어 있는 경우 SCC 리소스가 생성되지 않습니다.	true
이름	SCC의 이름을 지정합니다.	trident-protect-job
우선순위	SCC의 우선순위를 정의합니다. 우선순위 값이 높은 SCC는 값이 낮은 SCC보다 먼저 평가됩니다.	1

3. `sccconfig.yaml` 파일의 값을 적용합니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

이렇게 하면 기본값이 `sccconfig.yaml` 파일에 지정된 값으로 대체됩니다.

Trident Protect 헬름 차트의 추가 설정을 구성합니다

AutoSupport 설정 및 네임스페이스 필터링을 사용자 지정하여 특정 요구 사항을 충족할 수 있습니다. 다음 표는 사용 가능한 구성 매개변수를 설명합니다.

매개변수	유형	설명
autoSupport.프록시	문자열	NetApp AutoSupport 연결을 위한 프록시 URL을 구성합니다. 이 설정을 사용하여 지원 번들 업로드를 프록시 서버를 통해 라우팅합니다. 예: `http://my.proxy.url`.
autoSupport.insecure	boolean	AutoSupport 프록시 연결에 대해 `true`로 설정하면 TLS 검증을 건너뜁니다. 안전하지 않은 프록시 연결에만 사용하십시오. (기본값: `false`)
autoSupport.enabled	boolean	Trident Protect AutoSupport 번들 일일 업로드 기능을 활성화 또는 비활성화합니다. `false`로 설정하면 예약된 일일 업로드가 비활성화되지만 수동으로 지원 번들을 생성할 수는 있습니다. (기본값: `true`)

매개변수	유형	설명
restoreSkipNamespaceAnnotations	문자열	백업 및 복원 작업에서 제외할 네임스페이스 주석의 쉼표로 구분된 목록입니다. 주석을 기반으로 네임스페이스를 필터링할 수 있습니다.
restoreSkipNamespaceLabels	문자열	백업 및 복원 작업에서 제외할 네임스페이스 레이블을 쉼표로 구분하여 나열합니다. 레이블을 기준으로 네임스페이스를 필터링할 수 있습니다.

YAML 구성 파일 또는 명령줄 플래그를 사용하여 이러한 옵션을 구성할 수 있습니다.

YAML 파일 사용

단계

1. 구성 파일을 생성하고 이름을 지정하세요 `values.yaml`.
2. 생성한 파일에 사용자 지정하려는 구성 옵션을 추가하세요.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. `values.yaml` 파일에 올바른 값을 입력한 후 구성 파일을 적용합니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

CLI 플래그 사용

단계

1. `--set` 플래그와 함께 다음 명령을 사용하여 개별 매개변수를 지정하십시오.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Trident Protect Pod를 특정 노드로 제한합니다

Kubernetes nodeSelector 노드 선택 제약 조건을 사용하여 노드 레이블을 기반으로 Trident Protect Pod를 실행할 수 있는 노드를 제어할 수 있습니다. 기본적으로 Trident Protect는 Linux를 실행하는 노드로 제한됩니다. 필요에 따라 이러한 제약 조건을 추가로 사용자 지정할 수 있습니다.

단계

1. `nodeSelectorConfig.yaml`이라는 이름의 파일을 생성하세요.
2. 파일에 nodeSelector 옵션을 추가하고 환경 요구 사항에 따라 제한하도록 노드 레이블을 추가하거나 변경하도록 파일을 수정합니다. 예를 들어, 다음 파일에는 기본 OS 제한이 포함되어 있지만 특정 지역 및 앱 이름도 대상으로

지정합니다.

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. `nodeSelectorConfig.yaml` 파일의 값을 적용합니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

이렇게 하면 기본 제한 사항이 `nodeSelectorConfig.yaml` 파일에 지정한 제한 사항으로 대체됩니다.

Trident Protect 관리

Trident Protect 권한 부여 및 액세스 제어 관리

Trident Protect는 Kubernetes 모델의 역할 기반 액세스 제어(RBAC)를 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스와 해당 기본 서비스 계정을 제공합니다. 사용자 수가 많거나 특정 보안 요구 사항이 있는 조직의 경우 Trident Protect의 RBAC 기능을 활용하여 리소스 및 네임스페이스에 대한 액세스 제어를 더욱 세밀하게 수행할 수 있습니다.

클러스터 관리자는 항상 기본 `trident-protect` 네임스페이스의 리소스에 액세스할 수 있으며 다른 모든 네임스페이스의 리소스에도 액세스할 수 있습니다. 리소스 및 애플리케이션에 대한 액세스 제어를 위해 추가 네임스페이스를 생성하고 해당 네임스페이스에 리소스와 애플리케이션을 추가해야 합니다.

참고로, 기본 `trident-protect` 네임스페이스에서는 어떤 사용자도 애플리케이션 데이터 관리 CR을 생성할 수 없습니다. 애플리케이션 데이터 관리 CR은 애플리케이션 네임스페이스에서 생성해야 합니다(권장되는 방법은 해당 애플리케이션과 동일한 네임스페이스에 애플리케이션 데이터 관리 CR을 생성하는 것입니다).

> 관리자만 권한 있는 Trident Protect 사용자 지정 리소스 개체에 액세스할 수 있어야 합니다. 이러한 개체에는 다음이 포함됩니다.
>
> - **AppVault**: 버킷 자격 증명 데이터가 필요합니다
> - **AutoSupportBundle**: Trident Protect의 메트릭, 로그 및 기타 민감한 데이터를 수집합니다
> - **AutoSupportBundleSchedule**: 로그 수집 일정을 관리합니다
>
> 모범 사례로, RBAC를 사용하여 권한이 있는 개체에 대한 액세스를 클러스터 관리자로 제한하십시오.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 "Kubernetes RBAC 문서"을 참조하십시오.

서비스 계정에 대한 자세한 내용은 "Kubernetes 서비스 계정 문서"을 참조하십시오.

예: 두 사용자 그룹에 대한 액세스 관리

예를 들어, 한 조직에 클러스터 관리자, 엔지니어링 사용자 그룹, 마케팅 사용자 그룹이 있다고 가정해 보겠습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각각 자신의 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 구축하기 위해 다음과 같은 작업을 수행합니다.

1단계**:** 각 그룹의 리소스를 담을 네임스페이스를 생성합니다

네임스페이스를 생성하면 리소스를 논리적으로 분리하고 해당 리소스에 대한 액세스 권한을 더 효과적으로 제어할 수 있습니다.

단계

1. 엔지니어링 그룹을 위한 네임스페이스를 생성합니다.

```
kubectl create ns engineering-ns
```

2. 마케팅 그룹을 위한 네임스페이스를 생성하세요:

```
kubectl create ns marketing-ns
```

2단계**:** 각 네임스페이스의 리소스와 상호 작용할 새 서비스 계정 생성

새 네임스페이스를 생성할 때마다 기본 서비스 계정이 제공되지만, 향후 필요에 따라 그룹 간에 권한을 더욱 세분화할 수 있도록 각 사용자 그룹별로 서비스 계정을 생성해야 합니다.

단계

1. 엔지니어링 그룹을 위한 서비스 계정을 생성하세요:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 마케팅 그룹에 대한 서비스 계정 생성:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

3단계**:** 각 새 서비스 계정에 대한 시크릿 생성

서비스 계정 암호는 서비스 계정 인증에 사용되며, 유출된 경우 쉽게 삭제하고 다시 생성할 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 대한 암호를 생성합니다.

```
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 암호를 생성합니다.

```
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**: RoleBinding** 객체를 생성하여 **ClusterRole** 객체를 각 새 서비스 계정에 바인딩합니다

Trident Protect를 설치하면 기본 ClusterRole 오브젝트가 생성됩니다. 이 ClusterRole을 서비스 계정에 바인딩하려면 RoleBinding 오브젝트를 생성하고 적용할 수 있습니다.

단계

1. ClusterRole을 엔지니어링 서비스 계정에 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 확인하십시오.

```
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 확인합니다.

```
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 객체에 대한 액세스 권한을 부여해야 합니다.

단계

1. AppVault와 시크릿 조합 YAML 파일을 생성하여 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 네임스페이스 내 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 생성하고 적용합니다. 예를 들면 다음과 같습니다.

```
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 eng-user 사용자에게 권한을 바인딩합니다. 예시:

```
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 확인합니다.

   a. 모든 네임스페이스에 대한 AppVault 객체 정보를 검색하려고 시도합니다.

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   다음과 유사한 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 가져올 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 유사한 출력이 표시됩니다.

```
yes
```

결과

AppVault 권한을 부여받은 사용자는 승인된 AppVault 개체를 사용하여 애플리케이션 데이터 관리 작업을 수행할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 생성할 수 없어야 합니다.

Trident Protect 리소스를 모니터링하세요

kube-state-metrics, Prometheus 및 Alertmanager 오픈 소스 도구를 사용하여 Trident Protect로 보호되는 리소스의 상태를 모니터링할 수 있습니다.

kube-state-metrics 서비스는 Kubernetes API 통신에서 메트릭을 생성합니다. 이 서비스를 Trident Protect와 함께 사용하면 환경 내 리소스 상태에 대한 유용한 정보를 확인할 수 있습니다.

Prometheus는 kube-state-metrics에서 생성된 데이터를 수집하여 해당 객체에 대한 읽기 쉬운 정보로 제공하는 툴킷입니다. kube-state-metrics와 Prometheus를 함께 사용하면 Trident Protect로 관리하는 리소스의 상태를 모니터링할 수 있습니다.

Alertmanager는 Prometheus와 같은 도구에서 전송된 알림을 수집하여 구성한 대상으로 라우팅하는 서비스입니다.

> 이 단계에 포함된 구성 및 지침은 예시일 뿐이며, 사용자의 환경에 맞게 수정해야 합니다. 구체적인 지침 및 지원은 다음 공식 문서를 참조하십시오:
>
> - "kube-state-metrics 문서"
> - "Prometheus 문서"
> - "Alertmanager 문서"

1단계**:** 모니터링 도구 설치

Trident Protect에서 리소스 모니터링을 활성화하려면 kube-state-metrics, Prometheus 및 Alertmanager를 설치하고 구성해야 합니다.

kube-state-metrics 설치

Helm을 사용하여 kube-state-metrics를 설치할 수 있습니다.

단계

1. kube-state-metrics Helm 차트를 추가합니다. 예:

```
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 차트용 구성 파일을 생성합니다(예: `metrics-config.yaml`). 다음 예시 구성을 환경에 맞게 사용자 지정할 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 설치하세요. 예를 들면 다음과 같습니다.

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. "kube-state-metrics 사용자 정의 리소스 문서"의 지침에 따라 Trident Protect에서 사용하는 사용자 지정 리소스에 대한 메트릭을 생성하도록 kube-state-metrics를 구성합니다.

Prometheus 설치

https://prometheus.io/docs/prometheus/latest/installation/["Prometheus 문서"^]의 지침에 따라 Prometheus를 설치할 수 있습니다.

Alertmanager 설치

https://github.com/prometheus/alertmanager?tab=readme-ov-file#install["Alertmanager 문서"^]의 지침에 따라 Alertmanager를 설치할 수 있습니다.

2단계: 모니터링 도구가 함께 작동하도록 구성합니다

모니터링 도구를 설치한 후에는 함께 작동하도록 구성해야 합니다.

단계

1. kube-state-metrics를 Prometheus와 통합합니다. Prometheus 구성 파일(`prometheus.yaml`)을 편집하고 kube-state-metrics 서비스 정보를 추가합니다. 예를 들면 다음과 같습니다.

   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. Prometheus가 Alertmanager로 알림을 라우팅하도록 구성합니다. Prometheus 구성 파일(`prometheus.yaml`)을 편집하고 다음 섹션을 추가합니다.

prometheus.yaml: Alertmanager로 알림 전송

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

결과

이제 Prometheus는 kube-state-metrics에서 메트릭을 수집하고 Alertmanager로 알림을 보낼 수 있습니다. 이제 알림을 트리거하는 조건과 알림을 보낼 위치를 구성할 준비가 되었습니다.

3단계**:** 알림 및 알림 대상 구성

도구들을 연동하도록 구성한 후에는 어떤 유형의 정보가 알림을 발생시키는지, 그리고 알림을 어디로 보내야 하는지를 구성해야 합니다.

알림 예**:** 백업 실패

다음 예제는 백업 사용자 지정 리소스의 상태가 `Error`5초 이상 설정될 때 트리거되는 중요 경고를 정의합니다. 이 예제를 환경에 맞게 사용자 지정하고 이 YAML 코드 조각을 `prometheus.yaml` 구성 파일에 포함할 수 있습니다.

rules.yaml: 백업 실패에 대한 **Prometheus** 알림 정의

```
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."
```

Alertmanager가 다른 채널로 알림을 보내도록 구성합니다

Alertmanager를 구성하여 이메일, PagerDuty, Microsoft Teams 또는 기타 알림 서비스와 같은 다른 채널로 알림을 보낼 수 있습니다. 이를 위해서는 `alertmanager.yaml` 파일에 해당 구성을 지정하면 됩니다.

다음 예제는 Alertmanager가 Slack 채널로 알림을 보내도록 구성합니다. 이 예제를 사용자 환경에 맞게 사용자 지정하려면 `api_url` 키의 값을 사용자 환경에서 사용하는 Slack 웹훅 URL로 바꾸십시오.

alertmanager.yaml: Slack 채널로 알림 전송

```
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 번들을 생성합니다

Trident Protect를 사용하면 관리자가 관리 중인 클러스터 및 애플리케이션에 대한 로그, 메트릭 및 토폴로지 정보를 포함하여 NetApp 지원에 유용한 정보가 담긴 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 사용자 지정 리소스(CR) 파일을 사용하여 지원 번들을 NetApp 지원 사이트(NSS)에 업로드할 수 있습니다.

CR을 사용하여 지원 번들을 생성합니다

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml`).
2. 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.triggerType**: (필수) 지원 번들이 즉시 생성될지, 아니면 예약될지를 결정합니다. 예약된 번들 생성은 UTC 기준 오전 12시에 발생합니다. 가능한 값:
     - 예약됨
     - 수동
   - **spec.uploadEnabled**: (선택 사항) 지원 번들을 생성한 후 NetApp Support Site에 업로드할지 여부를 제어합니다. 지정하지 않으면 기본값은 `false`입니다. 가능한 값:
     - true
     - false(기본값)
   - **spec.dataWindowStart**: (선택 사항) 지원 번들에 포함된 데이터 기간이 시작되는 날짜 및 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않으면 기본값은 24시간 전입니다. 지정할 수 있는 가장 빠른 기간 시작 날짜는 7일 전입니다.

     YAML 예:

```
---
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. `trident-protect-support-bundle.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

CLI를 사용하여 지원 번들 생성

단계

1. 지원 번들을 생성하려면 괄호 안의 값을 사용자 환경 정보로 바꾸십시오. `trigger-type`은 번들이 즉시 생성되는지 아니면 예약된 시간에 생성되는지를 결정하며, `Manual` 또는 `Scheduled`일 수 있습니다.

기본 설정은 `Manual`입니다.

예를 들면 다음과 같습니다.

```
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 생성한 후에는 생성 진행 상황을 모니터링하고 로컬 시스템으로 가져올 수 있습니다.

단계

1. `status.generationState`이(가) `Completed` 상태에 도달할 때까지 기다립니다. 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. 완료된 AutoSupport 번들에서 복사 명령을 가져옵니다.

```
kubectl describe autosupportbundle trident-protect-support-bundle -n
trident-protect
```

출력에서 `kubectl cp` 명령을 찾아 실행하되, 대상 인수를 원하는 로컬 디렉터리로 바꾸십시오.

Trident Protect 업그레이드

Trident Protect를 최신 버전으로 업그레이드하면 새로운 기능이나 버그 수정 사항을 활용할 수 있습니다.

- 버전 24.10에서 업그레이드할 때 업그레이드 중에 실행되는 스냅샷이 실패할 수 있습니다. 이러한 실패는 수동 또는 예약된 향후 스냅샷 생성을 방해하지 않습니다. 업그레이드 중에 스냅샷이 실패하면 수동으로 새 스냅샷을 생성하여 애플리케이션을 보호할 수 있습니다.

  잠재적인 오류를 방지하려면 업그레이드 전에 모든 스냅샷 일정을 비활성화한 다음 업그레이드 후에 다시 활성화할 수 있습니다. 그러나 이렇게 하면 업그레이드 기간 동안 예약된 스냅샷이 누락될 수 있습니다.

- 프라이빗 레지스트리 설치의 경우, 타겟 버전에 필요한 Helm 차트 및 이미지가 프라이빗 레지스트리에 있는지 확인하고, 사용자 지정 Helm 값이 새 차트 버전과 호환되는지 확인하십시오. 자세한 내용은 "개인 레지스트리에서 Trident Protect를 설치하세요"를 참조하십시오.

Trident Protect를 업그레이드하려면 다음 단계를 수행하십시오.

단계

1. Trident Helm 리포지토리를 업데이트합니다.

```
helm repo update
```

2. Trident Protect CRD를 업그레이드합니다.

> 25.06 이전 버전에서 업그레이드하는 경우 이 단계가 필요합니다. CRD가 이제 Trident Protect Helm 차트에 포함되어 있기 때문입니다.

a. `trident-protect-crds`에서 `trident-protect`로 CRD 관리를 전환하려면 다음 명령을 실행합니다.

```
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. `trident-protect-crds` 차트의 Helm 시크릿을 삭제하려면 다음 명령을 실행하십시오.

> Helm을 사용하여 `trident-protect-crds` 차트를 제거하지 마십시오. 이렇게 하면 CRD 및 관련 데이터가 모두 삭제될 수 있습니다.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident Protect 업그레이드:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

> `--set logLevel=debug`을 업그레이드 명령에 추가하여 업그레이드 중 로깅 수준을 구성할 수 있습니다. 기본 로깅 수준은 `warn`입니다. 디버그 로깅은 NetApp 지원팀이 로그 수준 변경이나 문제 재현 없이 문제를 진단하는 데 도움이 되므로 문제 해결을 위해 권장됩니다.

애플리케이션 관리 및 보호

Trident Protect AppVault 객체를 사용하여 버킷을 관리하세요.

Trident Protect의 버킷 사용자 정의 리소스(CR)는 AppVault로 알려져 있습니다. AppVault 객체는 스토리지 버킷을 Kubernetes 워크플로에서 선언적으로 표현한 것입니다. AppVault CR에는 백업, 스냅샷, 복원 작업 및 SnapMirror 복제와 같은 보호 작업에 버킷을 사용하는 데 필요한 구성이 포함되어 있습니다. 관리자만 AppVaults를 생성할 수 있습니다.

애플리케이션에 대한 데이터 보호 작업을 수행할 때는 AppVault CR을 수동으로 또는 명령줄에서 생성해야 합니다. AppVault CR은 사용 환경에 따라 다르며, 이 페이지의 예제를 참고하여 AppVault CR을 생성할 수 있습니다.

> Trident Protect가 설치된 클러스터에 AppVault CR이 있는지 확인하십시오. AppVault CR이 없거나 액세스할 수 없는 경우 명령줄에 오류가 표시됩니다.

AppVault 인증 및 암호를 구성합니다

AppVault CR을 생성하기 전에 AppVault와 선택한 데이터 이동 도구가 공급자 및 관련 리소스에 인증할 수 있는지 확인하십시오.

Data Mover 리포지토리 암호

CR 또는 Trident Protect CLI 플러그인을 사용하여 AppVault 객체를 생성할 때 Restic 및 Kopia 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 시크릿을 지정하지 않으면 Trident Protect는 기본 암호를 사용합니다.

- AppVault CR을 수동으로 생성할 때는 **spec.dataMoverPasswordSecretRef** 필드를 사용하여 비밀 키를 지정하십시오.
- Trident Protect CLI를 사용하여 AppVault 객체를 생성할 때 `--data-mover-password-secret-ref` 인수를 사용하여 비밀 키를 지정하십시오.

데이터 이동 저장소 암호 보안 암호를 생성합니다

다음 예제를 사용하여 암호 비밀 키를 생성하세요. AppVault 객체를 생성할 때 Trident Protect가 이 비밀 키를 사용하여 데이터 이동 저장소와 인증하도록 설정할 수 있습니다.

> - 사용하는 데이터 이동 도구에 따라 해당 데이터 이동 도구에 대한 해당 암호만 포함하면 됩니다. 예를 들어 Restic을 사용하고 향후 Kopia를 사용할 계획이 없는 경우 시크릿을 생성할 때 Restic 암호만 포함하면 됩니다.
> - 암호를 안전한 곳에 보관하십시오. 동일한 클러스터 또는 다른 클러스터에서 데이터를 복원할 때 암호가 필요합니다. 클러스터 또는 `trident-protect` 네임스페이스가 삭제되면 암호 없이는 백업이나 스냅샷을 복원할 수 없습니다.

CR 사용

```
---
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
```

CLI 사용

```
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

Trident Protect를 사용하여 Amazon S3, Generic S3, "StorageGrid S3" 또는 "ONTAP S3"와 같은 S3 호환 스토리지에 액세스할 때는 제공하는 사용자 자격 증명에 버킷에 액세스하는 데 필요한 권한이 있는지 확인해야 합니다. 다음은 Trident Protect를 사용하여 액세스하는 데 필요한 최소 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 "Amazon S3 문서"의 예제를 참조하십시오.

Amazon S3(AWS) 인증을 위한 EKS Pod Identity

Trident Protect는 Kopia 데이터 이동 작업에서 EKS Pod Identity를 지원합니다. 이 기능을 통해 AWS 자격 증명을 Kubernetes 시크릿에 저장하지 않고도 S3 버킷에 안전하게 액세스할 수 있습니다.

Trident Protect를 사용한 **EKS Pod Identity** 요구 사항

Trident Protect에서 EKS Pod Identity를 사용하기 전에 다음 사항을 확인하십시오.

- EKS 클러스터에서 Pod Identity가 활성화되었습니다.
- 필요한 S3 버킷 권한이 있는 IAM 역할을 생성했습니다. 자세한 내용은 "S3 호환 스토리지 IAM 권한"를 참조하십시오.
- IAM 역할은 다음 Trident Protect 서비스 계정과 연결되어 있습니다.
  - `<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 "AWS EKS Pod Identity 문서"을 참조하십시오.

AppVault 구성 EKS Pod Identity를 사용하는 경우 `useIAM: true` 플래그를 사용하여 명시적인 자격 증명 대신 AppVault CR을 구성하십시오.

```
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 예

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 예시입니다. 다음 예시를 사용하여 각 클라우드 공급자의 자격 증명에 대한 키를 생성할 수 있습니다.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon 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
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

Generic 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
```

StorageGrid 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

다음은 각 공급자에 대한 AppVault 정의 예시입니다.

AppVault CR 예시

다음 CR 예제를 사용하여 각 클라우드 공급자에 대한 AppVault 객체를 생성할 수 있습니다.

- 선택적으로 Restic 및 Kopia 리포지토리 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 자세한 내용은 Data Mover 리포지토리 암호을 참조하십시오.
- Amazon S3(AWS) AppVault 객체의 경우 선택적으로 sessionToken을 지정할 수 있으며, 이는 Single Sign-On(SSO)을 인증에 사용하는 경우 유용합니다. 이 토큰은 클라우드 공급자를 위한 AppVault 키 생성 예에서 공급자용 키를 생성할 때 생성됩니다.
- S3 AppVault 객체의 경우 `spec.providerConfig.S3.proxyURL` 키를 사용하여 아웃바운드 S3 트래픽에 대한 이그레스 프록시 URL을 선택적으로 지정할 수 있습니다.

Google Cloud

```
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
```

Amazon 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, `providerCredentials` 섹션을 제거하고 `useIAM: true`를 `s3` 구성 아래에 추가할 수 있습니다.

Microsoft Azure

```
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
```

Generic 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
```

StorageGrid 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 예제

다음 CLI 명령 예제를 사용하여 각 공급자에 대한 AppVault CR을 생성할 수 있습니다.

- 선택적으로 Restic 및 Kopia 리포지토리 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 시크릿을 지정할 수 있습니다. 자세한 내용은 Data Mover 리포지토리 암호을 참조하십시오.
- S3 AppVault 객체의 경우 `--proxy-url <ip_address:port>` 인수를 사용하여 아웃바운드 S3 트래픽에 대한 이그레스 프록시 URL을 선택적으로 지정할 수 있습니다.

Google Cloud

```
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
```

Amazon 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
```

Microsoft Azure

```
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
```

Generic 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
```

StorageGrid 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
```

지원되는 `providerConfig.s3` 구성 옵션

S3 공급자 구성 옵션은 다음 표를 참조하십시오.

매개변수	설명	기본값	예
`providerConfig.s3.skipCertValidation`	SSL/TLS 인증서 확인을 비활성화합니다.	거짓	"true", "false"
`providerConfig.s3.secure`	S3 엔드포인트와의 안전한 HTTPS 통신을 활성화합니다.	true	"true", "false"
`providerConfig.s3.proxyURL`	S3에 연결하는 데 사용되는 프록시 서버의 URL을 지정하십시오.	None	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	SSL/TLS 검증을 위해 사용자 지정 루트 CA 인증서를 제공하십시오.	None	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	S3 버킷 액세스를 위한 IAM 인증을 활성화합니다. EKS Pod Identity에 적용됩니다.	거짓	참, 거짓

AppVault 정보를 확인하세요

Trident Protect CLI 플러그인을 사용하여 클러스터에 생성한 AppVault 객체에 대한 정보를 볼 수 있습니다.

단계

1. AppVault 객체의 내용을 보려면 다음을 수행하십시오.

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

예제 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 리소스의 AppVaultPath를 보려면 플래그 `--show-paths`를 사용하십시오.

   표의 첫 번째 열에 있는 클러스터 이름은 Trident Protect helm 설치 시 클러스터 이름이 지정된 경우에만 사용할 수 있습니다. 예를 들면 다음과 같습니다. `--set clusterName=production1`.

AppVault 제거

AppVault 개체는 언제든지 삭제할 수 있습니다.

> AppVault CR에서 `finalizers` 키를 AppVault 객체를 삭제하기 전에 제거하지 마십시오. 키를 제거하면 AppVault 버킷에 잔여 데이터가 남고 클러스터에 고아 리소스가 생성될 수 있습니다.

시작하기 전에

삭제하려는 AppVault에서 사용 중인 모든 스냅샷 및 백업 CR을 삭제했는지 확인하십시오.

Kubernetes CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거합니다. 제거할 AppVault 객체의 이름으로 `appvault-name`을(를) 바꿉니다.

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect CLI를 사용하여 **AppVault** 제거

1. AppVault 객체를 제거합니다. 제거할 AppVault 객체의 이름으로 `appvault-name`을(를) 바꿉니다.

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect를 사용하여 관리용 애플리케이션 정의

Trident Protect로 관리할 애플리케이션을 정의하려면 애플리케이션 CR과 연결된 AppVault CR을 생성하면 됩니다.

AppVault CR 생성

애플리케이션에 대한 데이터 보호 작업을 수행할 때 사용할 AppVault CR을 생성해야 하며, AppVault CR은 Trident Protect가 설치된 클러스터에 있어야 합니다. AppVault CR은 사용 환경에 따라 다릅니다. AppVault CR의 예는 "AppVault 사용자 지정 리소스."을 참조하십시오.

애플리케이션 정의

Trident Protect로 관리할 각 애플리케이션을 정의해야 합니다. 애플리케이션 CR을 수동으로 생성하거나 Trident Protect CLI를 사용하여 관리할 애플리케이션을 정의할 수 있습니다.

CR을 사용하여 애플리케이션 추가

단계

1. 타겟 애플리케이션 CR 파일을 생성합니다.
   a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `maria-app.yaml`).
   b. 다음 속성을 구성하십시오.
      - **metadata.name**: (*Required*) 애플리케이션 사용자 지정 리소스의 이름입니다. 보호 작업에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기억해 두십시오.
      - **spec.includedNamespaces**: (필수) 네임스페이스 및 레이블 선택기를 사용하여 애플리케이션에서 사용하는 네임스페이스와 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 목록에 반드시 포함되어야 합니다. 레이블 선택기는 선택 사항이며, 지정된 각 네임스페이스 내의 리소스를 필터링하는 데 사용할 수 있습니다.
      - **spec.includedClusterScopedResources**: (선택 사항) 이 속성을 사용하여 애플리케이션 정의에 포함할 클러스터 범위 리소스를 지정합니다. 이 속성을 사용하면 그룹, 버전, 종류 및 레이블을 기준으로 이러한 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 어노테이션은 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 동결되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 이 애플리케이션이 스냅샷 중에 파일 시스템에 쓸 수 있는지 여부를 지정합니다. true로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. false로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 동결됩니다. 지정되었지만 애플리케이션 정의에 가상 머신이 없는 경우 어노테이션은 무시됩니다. 지정하지 않으면 애플리케이션은 "글로벌 Trident Protect 동결 설정"를 따릅니다.

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 예:

+

```
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 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

    - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.

        - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
            - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

> ⓘ `resourceFilter`와 `labelSelector`가 모두 사용될 경우, `resourceFilter`가 먼저 실행된 다음 `labelSelector`가 결과 리소스에 적용됩니다.

예를 들면 다음과 같습니다.

```
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. 사용 환경에 맞는 애플리케이션 CR을 생성한 후 CR을 적용합니다. 예를 들면 다음과 같습니다.

```
kubectl apply -f maria-app.yaml
```

단계

1. 다음 예제 중 하나를 사용하여 애플리케이션 정의를 생성하고 적용하세요. 괄호 안의 값은 사용자 환경에 맞는 정보로 바꿔야 합니다. 예제에 표시된 인수를 쉼표로 구분한 목록을 사용하여 네임스페이스와 리소스를 애플리케이션 정의에 포함할 수 있습니다.

   앱을 생성할 때 스냅샷 중에 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정하는 어노테이션을 선택적으로 사용할 수 있습니다. 이는 KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 고정되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 어노테이션을 `true`로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. 어노테이션을 `false`로 설정하면 애플리케이션은 전역 설정을 무시하고 스냅샷 중에 파일 시스템이 고정됩니다. 어노테이션을 사용했지만 애플리케이션 정의에 가상 머신이 없는 경우 어노테이션은 무시됩니다. 어노테이션을 사용하지 않으면 애플리케이션은 "글로벌 Trident Protect 동결 설정"를 따릅니다.

CLI를 사용하여 애플리케이션을 생성할 때 주석을 지정하려면 `--annotation` 플래그를 사용할 수 있습니다.

- 애플리케이션을 생성하고 파일 시스템 동결 동작에 대한 글로벌 설정을 사용하십시오.

```
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 고정 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">
```

`--resource-filter-include` 및 `--resource-filter-exclude` 플래그를 사용하여 다음 예와 같이 그룹, 종류, 버전, 레이블, 이름 및 네임스페이스와 같은 `resourceSelectionCriteria`을(를) 기반으로 리소스를 포함하거나 제외할 수 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 애플리케이션을 보호하세요.

Trident Protect에서 관리하는 모든 앱은 자동 보호 정책을 사용하거나 임시로 스냅샷 및 백업을 생성하여 보호할 수 있습니다.

Trident Protect를 구성하면 데이터 보호 작업 중에 파일 시스템을 동결 및 해제할 수 있습니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".

온디맨드 스냅샷 생성

언제든지 필요에 따라 스냅샷을 생성할 수 있습니다.

클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 스냅샷 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠(메타데이터)를 저장할 AppVault의 이름입니다.
   - **spec.reclaimPolicy**: (선택 사항) 스냅샷 CR이 삭제될 때 스냅샷의 AppArchive에 어떤 일이 발생하는지 정의합니다. 즉, `Retain`로 설정하더라도 스냅샷은 삭제됩니다. 유효한 옵션:
     - `Retain` (기본값)
     - `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. `trident-protect-snapshot-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

CLI를 사용하여 스냅샷 생성

단계

1. 스냅샷을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꾸세요. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

온디맨드 백업 생성

언제든지 앱을 백업할 수 있습니다.

클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

시작하기 전에

장시간 실행되는 s3 백업 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 백업 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR을 사용하여 백업 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠를 저장할 AppVault의 이름입니다.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia` (기본값)
   - **spec.reclaimPolicy**: (선택 사항) 백업이 클레임에서 해제될 때 발생하는 상황을 정의합니다. 가능한 값:
     - `Delete`
     - `Retain` (기본값)
   - **spec.snapshotRef**: (선택 사항): 백업 소스로 사용할 스냅샷의 이름입니다. 지정하지 않으면 임시 스냅샷이 생성되어 백업됩니다.

     YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. `trident-protect-backup-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-backup-cr.yaml
```

CLI를 사용하여 백업 생성

단계

1. 백업을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꿔주세요. 예를 들면 다음과 같습니다.

```
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 `--full-backup` 플래그를 사용하여 백업을 비증분 방식으로 수행할지 여부를 지정할 수 있습니다. 기본적으로 모든 백업은 증분 방식으로 수행됩니다. 이 플래그를 사용하면 백업이 비증분 방식으로 수행됩니다. 복원과 관련된 위험을 최소화하려면 정기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 좋습니다.

지원되는 백업 주석

다음 표에서는 백업 CR을 생성할 때 사용할 수 있는 주석에 대해 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/full-backup	문자열	백업을 비증분 백업으로 수행할지 여부를 지정합니다. `true`로 설정하여 비증분 백업을 생성합니다. 복원과 관련된 위험을 최소화하려면 주기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 좋습니다.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30분"
protect.trident.netapp.io/volume-snapshots-created-timeout	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다.	"1200" (20분)

데이터 보호 일정 생성

보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 스냅샷과 백업 생성 주기를 시간별, 일별, 주별, 월별로 설정할 수 있으며, 보존할 백업 복사본의 개수를 지정할 수도 있습니다. full-backup-rule 어노테이션을 사용하면 증분 백업이 아닌 전체 백업을 예약할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 전체 백업을 주기적으로 수행하고 그 사이에 증분 백업을 수행하면 복원과 관련된 위험을 줄이는 데 도움이 됩니다.

- `backupRetention`을 0으로 설정하고 `snapshotRetention`을 0보다 큰 값으로 설정하여 스냅샷 전용 스케줄을 생성할 수 있습니다. `snapshotRetention`을 0으로 설정하면 예약된 백업에서 여전히 스냅샷이 생성되지만 이는 임시적이며 백업이 완료된 후 즉시 삭제됩니다.
- 클러스터 범위 리소스는 애플리케이션 정의에서 명시적으로 참조되거나 애플리케이션 네임스페이스를 참조하는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 일정 생성

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-schedule-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.dataMover**: (선택 사항) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
     - `Restic`
     - `Kopia` (기본값)
   - **spec.applicationRef**: 백업할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠를 저장할 AppVault의 이름입니다.
   - **spec.backupRetention**: (필수) 보존할 백업 개수입니다. 0으로 설정하면 백업을 생성하지 않고 스냅샷만 생성합니다.
   - **spec.backupReclaimPolicy**: (선택 사항) 보존 기간 동안 백업 CR이 삭제될 경우 백업 데이터에 발생하는 작업을 결정합니다. 보존 기간 이후에는 백업이 항상 삭제됩니다. 가능한 값(대소문자 구분):
     - `Retain` (기본값)
     - `Delete`
   - **spec.snapshotRetention**: (필수) 보존할 스냅샷 개수입니다. 0으로 설정하면 스냅샷을 생성하지 않습니다.
   - **spec.snapshotReclaimPolicy**: (선택 사항) 보존 기간 동안 스냅샷 CR이 삭제될 경우 스냅샷에 발생하는 상황을 결정합니다. 보존 기간이 지나면 스냅샷은 항상 삭제됩니다. 가능한 값(대소문자 구분):
     - `Retain`
     - `Delete` (기본값)
   - **spec.granularity**: 스케줄이 실행될 빈도입니다. 가능한 값과 필수 관련 필드는 다음과 같습니다.
     - `Hourly` (지정해야 합니다 `spec.minute`)
     - `Daily` (`spec.minute` 및 `spec.hour`을 지정해야 함)
     - `Weekly`(지정해야 함 `spec.minute, spec.hour`, `spec.dayOfWeek`)
     - `Monthly`(, `spec.minute, spec.hour` 및 `spec.dayOfMonth`를 지정해야 합니다)
     - `Custom`
   - **spec.dayOfMonth**: (선택 사항) 스케줄이 실행될 날짜(1~31일)입니다. 세분성이 `Monthly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.dayOfWeek**: (선택 사항) 스케줄을 실행할 요일(0 - 7)입니다. 0 또는 7 값은 일요일을 나타냅니다. 세분성이 `Weekly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.hour**: (선택 사항) 스케줄이 실행될 시간(0 - 23)입니다. 이 필드는 세분성이 `Daily`, `Weekly` 또는 `Monthly`로 설정된 경우 필수입니다. 값은 문자열로 제공해야 합니다.
   - **spec.minute**: (선택 사항) 스케줄이 실행되어야 하는 시간의 분(0 - 59)입니다. 세분성이 `Hourly`,

`Daily`, `Weekly` 또는 `Monthly`로 설정된 경우 이 필드는 필수입니다. 값은 문자열로 제공해야 합니다.

백업 및 스냅샷 스케줄에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

스냅샷 전용 스케줄의 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. `trident-protect-schedule-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

CLI를 사용하여 스케줄 생성

단계

1. 보호 일정을 생성하고 괄호 안의 값을 사용자 환경 정보로 대체하십시오. 예를 들면 다음과 같습니다.

   `tridentctl-protect create schedule --help`을 사용하여 이 명령에 대한 자세한 도움말 정보를 볼 수 있습니다.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_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> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

   다음 플래그를 사용하면 일정을 더욱 세밀하게 제어할 수 있습니다.

   - 전체 백업 예약: `--full-backup-rule` 플래그를 사용하여 증분 방식이 아닌 전체 백업을 예약합니다. 이 플래그는 `--granularity Daily`에서만 작동합니다. 가능한 값:
     - `Always`: 매일 전체 백업을 생성하세요.
     - 특정 요일: 쉼표로 구분하여 하나 이상의 요일을 지정합니다(예: `"Monday,Thursday"`). 유효한 값: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

       `--full-backup-rule` 플래그는 시간별, 주별 또는 월별 세분화에서는 작동하지 않습니다.

   - 스냅샷 전용 일정: `--backup-retention 0`을(를) 설정하고 `--snapshot-retention`에 대해 0보다 큰 값을 지정하십시오.

지원되는 스케줄 주석

다음 표에서는 스케줄 CR을 생성할 때 사용할 수 있는 주석에 대해 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/full-backup-rule	문자열	전체 백업 예약 규칙을 지정합니다. `Always`로 설정하여 정기적인 전체 백업을 수행하거나 필요에 따라 사용자 지정할 수 있습니다. 예를 들어, 일 단위로 예약하는 경우 전체 백업이 수행될 요일을 지정할 수 있습니다(예: `"Monday,Thursday"`). 유효한 요일 값은 Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday입니다. 이 어노테이션은 `granularity`가 `Daily`로 설정된 예약에만 사용할 수 있습니다.	설정되지 않음(모든 백업은 증분)
protect.trident.netapp.io/snapshot-completion-timeout	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30분"
protect.trident.netapp.io/volume-snapshots-created-timeout	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다.	"1200" (20분)

스냅샷 삭제

더 이상 필요하지 않은 예약된 스냅샷 또는 필요 시 스냅샷을 삭제합니다.

단계

1. 스냅샷과 연결된 스냅샷 CR을 제거합니다.

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

백업 삭제

더 이상 필요하지 않은 예약된 백업 또는 온디맨드 백업을 삭제합니다.

> 객체 스토리지에서 모든 백업 데이터를 제거하려면 reclaim 정책이 `Delete`로 설정되어 있는지 확인하십시오. 정책의 기본 설정은 의도하지 않은 데이터 손실을 방지하기 위해 `Retain`입니다. 정책을 `Delete`로 변경하지 않으면 백업 데이터가 객체 스토리지에 남아 있으므로 수동으로 삭제해야 합니다.

단계

1. 백업과 연결된 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 대괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

```
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 차지하지 않습니다.

시작하기 전에

다음 사항을 확인하십시오.

- Trident Protect를 설치했습니다.
- Trident Protect에서 애플리케이션을 정의했습니다. 이 절차를 완료할 때까지 이 애플리케이션의 보호 기능이 제한됩니다.
- `azure-netapp-files`을(를) 스토리지 백엔드의 기본 스토리지 클래스로 선택했습니다.

구성 단계를 보려면 펼치세요

1. ANF 볼륨이 Trident 24.10으로 업그레이드하기 전에 생성된 경우 Trident에서 다음을 수행하십시오.

   a. 애플리케이션과 연결된 azure-netapp-files 기반의 각 PV에 대해 스냅샷 디렉터리를 활성화합니다.

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. 각 연결된 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인합니다.

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   응답:

   ```
   snapshotDirectory: "true"
   ```

   +
   스냅샷 디렉터리가 활성화되지 않은 경우, Trident Protect는 일반 백업 기능을 선택하며, 이 기능은 백업 프로세스 중에 용량 풀의 공간을 일시적으로 사용합니다. 이 경우 백업 대상 볼륨 크기의 임시 볼륨을 생성할 수 있도록 용량 풀에 충분한 공간이 확보되어 있는지 확인하십시오.

결과

이 애플리케이션은 Trident Protect를 사용하여 백업 및 복원할 준비가 되어 있습니다. 각 PVC는 다른 애플리케이션에서도 백업 및 복원에 사용할 수 있습니다.

애플리케이션 복원

Trident Protect를 사용하여 애플리케이션을 복원하세요

Trident Protect를 사용하면 스냅샷 또는 백업에서 애플리케이션을 복원할 수 있습니다. 기존 스냅샷에서 복원하면 동일한 클러스터에 애플리케이션을 복원할 때 더 빠릅니다.

- 애플리케이션을 복원하면 해당 애플리케이션에 대해 구성된 모든 실행 후크가 애플리케이션과 함께 복원됩니다. 복원 후 실행 후크가 있는 경우 복원 작업의 일부로 자동으로 실행됩니다.
- qtree 볼륨의 경우 백업에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 것이 지원됩니다. 그러나 스냅샷에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 것은 qtree 볼륨에서 지원되지 않습니다.
- 고급 설정을 사용하여 복원 작업을 사용자 지정할 수 있습니다. 자세한 내용은 "고급 Trident Protect 복원 설정을 사용하십시오"를 참조하십시오.

BackupRestore CR을 사용하여 다른 네임스페이스로 백업을 복원하면 Trident Protect는 새 네임스페이스에 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 생성합니다. 복원된 애플리케이션을 보호하려면 필요에 따라 백업 또는 스냅샷을 생성하거나 보호 일정을 설정하십시오.

- 기존 리소스가 있는 다른 네임스페이스로 백업을 복원해도 백업에 포함된 리소스와 이름이 같은 리소스는 변경되지 않습니다. 백업의 모든 리소스를 복원하려면 대상 네임스페이스를 삭제하고 다시 생성하거나 새 네임스페이스로 백업을 복원해야 합니다.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 콘텐츠가 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 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   > Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.

- **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

  예를 들면 다음과 같습니다.

```
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. `trident-protect-backup-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI 사용

단계

1. 백업을 다른 네임스페이스로 복원하고, 괄호 안의 값을 사용자 환경 정보로 바꿉니다. `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식으로 올바른 타겟 네임스페이스에 매핑합니다. 예:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

원래 네임스페이스로 백업에서 복원합니다

언제든지 백업을 원래 네임스페이스로 복원할 수 있습니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-ipr-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 백업 콘텐츠가 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 콘텐츠가 저장되는 AppVault의 이름입니다.

     예를 들면 다음과 같습니다.

```
---
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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

  예를 들면 다음과 같습니다.

```
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. `trident-protect-backup-ipr-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

CLI 사용

단계

1. 원래 네임스페이스에 백업을 복원하고, 괄호 안의 값을 사용자 환경 정보로 바꿉니다. `backup` 인수는 `<namespace>/<name>` 형식으로 네임스페이스와 백업 이름을 사용합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

원래 클러스터에 문제가 발생한 경우 백업을 다른 클러스터로 복원할 수 있습니다.

- Kopia를 데이터 이동 도구로 사용하여 백업을 복원할 때 CR 또는 CLI를 사용하여 Kopia에서 사용하는 임시 스토리지의 동작을 제어하는 주석을 선택적으로 지정할 수 있습니다. 구성할 수 있는 옵션에 대한 자세한 내용은 "Kopia 문서"을 참조하십시오. Trident Protect CLI를 사용하여 주석을 지정하는 방법에 대한 자세한 내용은 `tridentctl-protect create --help` 명령을 사용하십시오.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

다음 사전 요구 사항이 충족되는지 확인하십시오.

- 타겟 클러스터에 Trident Protect가 설치되어 있습니다.
- 타겟 클러스터는 백업이 저장된 소스 클러스터와 동일한 AppVault의 버킷 경로에 액세스할 수 있습니다.
- `tridentctl-protect get appvaultcontent` 명령을 실행할 때 로컬 환경에서 AppVault CR에 정의된 오브젝트 스토리지 버킷에 연결할 수 있는지 확인하십시오. 네트워크 제한으로 인해 액세스할 수 없는 경우 타겟 클러스터의 Pod 내에서 Trident Protect CLI를 실행하십시오.
- 장시간 소요되는 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.
  - 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
  - AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS 문서"를 참조하십시오.

단계

1. Trident Protect CLI 플러그인을 사용하여 타겟 클러스터에서 AppVault CR의 가용성을 확인합니다.

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

애플리케이션 복원에 사용할 네임스페이스가 타겟 클러스터에 있는지 확인합니다.

2. 타겟 클러스터에서 사용 가능한 AppVault의 백업 콘텐츠를 확인합니다.

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

이 명령을 실행하면 AppVault에서 사용 가능한 백업이 표시되며, 여기에는 백업이 생성된 클러스터, 해당 애플리케이션 이름, 타임스탬프 및 아카이브 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 아카이브 경로를 사용하여 타겟 클러스터에 애플리케이션을 복원합니다.

CR 사용

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-backup-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.appArchivePath**: 백업 콘텐츠가 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 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     예를 들면 다음과 같습니다.

```
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. `trident-protect-backup-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI 사용

1. 다음 명령을 사용하여 애플리케이션을 복원하고 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. namespace-mapping 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 source1:dest1,source2:dest2 형식으로 올바른 타겟 네임스페이스에 매핑합니다. 예를 들면 다음과 같습니다.

```
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

스냅샷에서 사용자 지정 리소스(CR) 파일을 사용하여 다른 네임스페이스 또는 원래 소스 네임스페이스로 데이터를 복원할 수 있습니다. SnapshotRestore CR을 사용하여 스냅샷을 다른 네임스페이스로 복원하면 Trident Protect는 새 네임스페이스에 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 생성합니다. 복원된 애플리케이션을 보호하려면 온디맨드 백업 또는 스냅샷을 생성하거나 보호 일정을 설정하십시오.

- SnapshotRestore는 `spec.storageClassMapping` 속성을 지원하지만, 소스 및 타겟 스토리지 클래스가 동일한 스토리지 백엔드를 사용하는 경우에만 가능합니다. 다른 스토리지 백엔드를 사용하는 `StorageClass`로 복원을 시도하면 복원 작업이 실패합니다.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 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 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

   > Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.

- **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

  예를 들면 다음과 같습니다.

```
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. `trident-protect-snapshot-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI 사용

단계

1. 스냅샷을 다른 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

   - `snapshot` 인수는 `<namespace>/<name>` 형식의 네임스페이스와 스냅샷 이름을 사용합니다.
   - `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식의 올바른 대상 네임스페이스에 매핑합니다.

예를 들면 다음과 같습니다.

```
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 복원합니다

언제든지 스냅샷을 원래 네임스페이스로 복원할 수 있습니다.

시작하기 전에

장시간 소요되는 s3 복원 작업의 경우 AWS 세션 토큰 만료 기간이 충분한지 확인하십시오. 복원 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 "AWS API 문서"을 참조하십시오.
- AWS 리소스에 대한 자격 증명 관련 자세한 내용은 "AWS IAM 문서"를 참조하십시오.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-snapshot-ipr-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appVaultRef**: (필수) 스냅샷 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 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) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하십시오.

> Trident Protect는 사용자가 선택한 리소스와의 연관성 때문에 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 해당 리소스에 연결된 Pod가 있는 경우 Trident Protect는 연결된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `Include` 또는 `Exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

  예를 들면 다음과 같습니다.

```
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. `trident-protect-snapshot-ipr-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

CLI 사용

단계

1. 스냅샷을 원래 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경 정보로 바꾸십시오. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

복원 작업의 상태를 확인합니다

명령줄을 사용하여 진행 중이거나 완료되었거나 실패한 복원 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 복원 작업의 상태를 검색하고 대괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

고급 **Trident Protect** 복원 설정을 사용하십시오

주석, 네임스페이스 설정, 스토리지 옵션과 같은 고급 설정을 사용하여 특정 요구 사항에 맞게 복원 작업을 사용자 지정할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 조정됩니다. 대상 네임스페이스에 존재하지 않는 소스 네임스페이스의 레이블 또는 주석은 추가되며, 이미 존재하는 레이블 또는 주석은 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 존재하는 레이블 또는 주석은 변경되지 않습니다.

Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 어노테이션의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 어노테이션은 복원된 Pod가 OpenShift 보안 컨텍스트 제약 조건(SCC)에 정의된 적절한 권한 및 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 "OpenShift 보안 컨텍스트 제약 조건 문서"을(를) 참조하십시오.

복원 또는 장애 조치 작업을 수행하기 전에 Kubernetes 환경 변수 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`를 설정하여 대상 네임스페이스의 특정 어노테이션이 덮어쓰여지는 것을 방지할 수 있습니다. 예를 들면 다음과 같습니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

복원 또는 페일오버 작업을 수행할 때 `restoreSkipNamespaceAnnotations` 및 `restoreSkipNamespaceLabels`에 지정된 네임스페이스 어노테이션 및 레이블은 복원 또는 페일오버 작업에서 제외됩니다. 초기 Helm 설치 시 이러한 설정이 구성되어 있는지 확인하십시오. 자세한 내용은 "Trident Protect 헬름 차트의 추가 설정을 구성합니다"를 참조하십시오.

`--create-namespace` 플래그와 함께 `Helm`을 사용하여 소스 애플리케이션을 설치한 경우 `name` 레이블 키에 특별한 처리가 적용됩니다. 복원 또는 페일오버 프로세스 중에 `Trident Protect`는 이 레이블을 대상 네임스페이스로 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제는 서로 다른 어노테이션과 레이블을 가진 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태와 어노테이션 및 레이블이 대상 네임스페이스에서 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복원 또는 페일오버 작업 전에

다음 표는 복원 또는 페일오버 작업 전 예시 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • 컴플라이언스=hipaa • name=ns-1
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후 예시 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고, 일부는 덮어쓰여졌으며, `name` 레이블은 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • 컴플라이언스=hipaa • environment=production • role=database

지원되는 필드

이 섹션에서는 복원 작업에 사용할 수 있는 추가 필드에 대해 설명합니다.

스토리지 클래스 매핑

`spec.storageClassMapping` 속성은 소스 애플리케이션에 있는 스토리지 클래스를 대상 클러스터의 새 스토리지 클래스로 매핑하는 방법을 정의합니다. 이 속성은 스토리지 클래스가 다른 클러스터 간에 애플리케이션을 마이그레이션하거나 `BackupRestore` 작업의 스토리지 백엔드를 변경할 때 사용할 수 있습니다.

예:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

지원되는 주석

이 섹션에서는 시스템의 다양한 동작을 구성하는 데 지원되는 어노테이션 목록을 제공합니다. 사용자가 어노테이션을 명시적으로 설정하지 않으면 시스템은 기본값을 사용합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/data-mover-timeout-sec	문자열	데이터 이동기 작동이 정지될 수 있는 최대 허용 시간(초)입니다.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	문자열	Kopia 콘텐츠 캐시의 최대 크기 제한(메가바이트)입니다.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 `Bound` 단계에 도달하기까지 기다리는 최대 시간(초)입니다. 이 시간이 지나면 작업이 실패합니다. 모든 복원 CR 유형(BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore)에 적용됩니다. 스토리지 백엔드 또는 클러스터에서 더 많은 시간이 필요한 경우 더 높은 값을 사용하십시오.	"1200" (20분)

NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션을 복제합니다

Trident Protect를 사용하면 NetApp SnapMirror 기술의 비동기 복제 기능을 활용하여 동일한 클러스터 내에서 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 한 스토리지 백엔드에서 다른 스토리지 백엔드로 복제할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석은 소스 네임스페이스의 레이블과 주석과 일치하도록 조정됩니다. 대상 네임스페이스에 존재하지 않는 소스 네임스페이스의 레이블 또는 주석은 추가되며, 이미 존재하는

레이블 또는 주석은 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 존재하는 레이블 또는 주석은 변경되지 않습니다.

Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 어노테이션의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 어노테이션은 복원된 Pod가 OpenShift 보안 컨텍스트 제약 조건(SCC)에 정의된 적절한 권한 및 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 "OpenShift 보안 컨텍스트 제약 조건 문서"을(를) 참조하십시오.

복원 또는 장애 조치 작업을 수행하기 전에 Kubernetes 환경 변수 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`를 설정하여 대상 네임스페이스의 특정 어노테이션이 덮어쓰여지는 것을 방지할 수 있습니다. 예를 들면 다음과 같습니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

복원 또는 페일오버 작업을 수행할 때 `restoreSkipNamespaceAnnotations` 및 `restoreSkipNamespaceLabels`에 지정된 네임스페이스 어노테이션 및 레이블은 복원 또는 페일오버 작업에서 제외됩니다. 초기 Helm 설치 시 이러한 설정이 구성되어 있는지 확인하십시오. 자세한 내용은 "Trident Protect 헬름 차트의 추가 설정을 구성합니다"를 참조하십시오.

`--create-namespace` 플래그와 함께 `Helm`을 사용하여 소스 애플리케이션을 설치한 경우 `name` 레이블 키에 특별한 처리가 적용됩니다. 복원 또는 페일오버 프로세스 중에 `Trident Protect`는 이 레이블을 대상 네임스페이스로 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 없이 대상 네임스페이스로 복사됩니다.

예

다음 예제는 서로 다른 어노테이션과 레이블을 가진 소스 및 대상 네임스페이스를 보여줍니다. 작업 전후의 대상 네임스페이스 상태와 어노테이션 및 레이블이 대상 네임스페이스에서 어떻게 결합되거나 덮어쓰여지는지 확인할 수 있습니다.

복원 또는 페일오버 작업 전에

다음 표는 복원 또는 페일오버 작업 전 예시 소스 및 대상 네임스페이스의 상태를 보여줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • 컴플라이언스=hipaa • name=ns-1

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

복원 작업 후

다음 표는 복원 또는 장애 조치 작업 후 예시 대상 네임스페이스의 상태를 보여줍니다. 일부 키가 추가되었고, 일부는 덮어쓰여졌으며, `name` 레이블은 대상 네임스페이스와 일치하도록 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • 컴플라이언스=hipaa • environment=production • role=database

Trident Protect를 구성하면 데이터 보호 작업 중에 파일 시스템을 동결 및 해제할 수 있습니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".

페일오버 및 역방향 작업 중 실행 후크

AppMirror 관계를 사용하여 애플리케이션을 보호할 때 페일오버 및 역방향 작업 중에 알아야 할 실행 후크와 관련된 특정 동작이 있습니다.

- 장애 조치 시 실행 후크는 소스 클러스터에서 타겟 클러스터로 자동으로 복사됩니다. 수동으로 다시 생성할 필요가 없습니다. 장애 조치가 완료되면 애플리케이션에 실행 후크가 존재하며 관련 작업이 수행될 때 실행됩니다.
- 역방향 또는 역방향 재동기화 중에 애플리케이션의 기존 실행 후크가 제거됩니다. 소스 애플리케이션이 타겟 애플리케이션이 되면 이러한 실행 후크는 유효하지 않으며 실행을 방지하기 위해 삭제됩니다.

실행 후크에 대한 자세한 내용은 "Trident Protect 실행 후크 관리"을(를) 참조하십시오.

복제 관계를 설정합니다

복제 관계를 설정하는 과정은 다음과 같습니다.

- Trident Protect에서 앱 스냅샷(앱의 Kubernetes 리소스와 앱의 각 볼륨에 대한 볼륨 스냅샷 포함)을 생성할 빈도를 선택합니다
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 생성할 시간 설정

단계

1. 소스 클러스터에서 소스 애플리케이션용 AppVault를 생성합니다. 사용 중인 스토리지 공급자에 따라 "AppVault 사용자 지정 리소스"의 예제를 환경에 맞게 수정하세요.

CR을 사용하여 **AppVault**를 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-primary-source.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) AppVault 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
- **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. bucketName 및 공급자에 필요한 기타 세부 정보를 선택하십시오. 복제 관계에 필요한 다른 CR 파일에서 이러한 값을 참조하므로 선택한 값을 기록해 두십시오. 다른 공급자를 사용하는 AppVault CR의 예는 "AppVault 사용자 지정 리소스"을 참조하십시오.
- **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
  - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀 키에서 가져와야 함을 나타냅니다.
    - **key**: (필수) 선택할 시크릿의 유효한 키입니다.
    - **name**: (필수) 이 필드의 값을 포함하는 시크릿의 이름입니다. 동일한 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 서비스(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure)를 지정합니다. 가능한 값:
  - aws
  - Azure
  - gcp
  - generic-s3
  - ONTAP S3
  - storagegrid-s3

c. `trident-protect-appvault-primary-source.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

CLI를 사용하여 **AppVault**를 생성합니다

a. AppVault를 생성하고 괄호 안의 값을 사용자 환경 정보로 바꾸세요.

```
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 생성합니다.

CR을 사용하여 소스 애플리케이션을 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-app-source.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 애플리케이션 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
- **spec.includedNamespaces**: (필수) 네임스페이스 및 관련 레이블 배열입니다. 네임스페이스 이름을 사용하고 선택적으로 레이블을 사용하여 네임스페이스의 범위를 좁혀 여기에 나열된 네임스페이스에 존재하는 리소스를 지정합니다. 애플리케이션 네임스페이스는 이 배열에 반드시 포함되어야 합니다.

  YAML 예:

```
---
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. `trident-protect-app-source.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

CLI를 사용하여 소스 애플리케이션 생성

a. 소스 애플리케이션을 생성합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. 선택적으로 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성할 수 있습니다. 이 스냅샷은 타겟 클러스터의 애플리케이션에 대한 기반으로 사용됩니다. 이 단계를 건너뛰면 최신 스냅샷을 얻기 위해 다음 예약된 스냅샷이 실행될 때까지 기다려야 합니다. 온디맨드 스냅샷을 생성하려면 "온디맨드 스냅샷 생성"을 참조하십시오.
4. 소스 클러스터에서 복제 일정 CR을 생성합니다.

아래 제공된 일정 외에도, 피어링된 ONTAP 클러스터 간의 공통 스냅샷을 유지하기 위해 7일의 보존 기간을 가진 별도의 일일 스냅샷 일정을 생성하는 것이 좋습니다. 이렇게 하면 스냅샷을 최대 7일 동안 사용할 수 있지만, 보존 기간은 사용자 요구 사항에 따라 사용자 지정할 수 있습니다.

장애 조치가 발생할 경우, 시스템은 최대 7일 동안 이러한 스냅샷을 사용하여 역방향 작업을 수행할 수 있습니다. 이 방식을 통해 역방향 프로세스는 모든 데이터가 아닌 마지막 스냅샷 이후 변경된 내용만 전송되므로 더 빠르고 효율적으로 진행됩니다.

애플리케이션에 대한 기존 스케줄이 원하는 보존 요구 사항을 이미 충족하는 경우 추가 스케줄이 필요하지 않습니다.

CR을 사용하여 복제 일정을 생성합니다

a. 소스 애플리케이션에 대한 복제 일정을 생성합니다.

i. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-schedule.yaml`).

ii. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 스케줄 사용자 지정 리소스의 이름입니다.
- **spec.appVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault에 있는 metadata.name 필드와 일치해야 합니다.
- **spec.applicationRef**: (필수) 이 값은 소스 애플리케이션 CR의 metadata.name 필드와 일치해야 합니다.
- **spec.backupRetention**: (필수) 이 필드는 필수이며 값을 0으로 설정해야 합니다.
- **spec.enabled**: true로 설정해야 합니다.
- **spec.granularity**: `Custom`로 설정해야 합니다.
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 정의합니다.
- **spec.snapshotRetention**: 2로 설정해야 합니다.

YAML 예:

```
---
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. `trident-protect-schedule.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

CLI를 사용하여 복제 스케줄 생성

a. 복제 일정을 생성하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

```
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. 타겟 클러스터에서 소스 클러스터에 적용한 AppVault CR과 동일한 소스 애플리케이션 AppVault CR을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-primary-destination.yaml`).
6. CR 적용:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. 타겟 클러스터의 타겟 애플리케이션에 대한 타겟 AppVault CR을 생성합니다. 스토리지 공급자에 따라 "AppVault 사용자 지정 리소스"의 예제를 환경에 맞게 수정하십시오.
   a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-appvault-secondary-destination.yaml`).
   b. 다음 속성을 구성하십시오.
      - **metadata.name**: (필수) AppVault 사용자 지정 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일에서 이 값을 참조하므로 선택한 이름을 기록해 두십시오.
      - **spec.providerConfig**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. `bucketName`및 공급자에 필요한 기타 세부 정보를 선택하십시오. 복제 관계에 필요한 다른 CR 파일에서 이러한 값을 참조하므로 선택한 값을 기록해 두십시오. 다른 공급자를 사용하는 AppVault CR의 예는 "AppVault 사용자 지정 리소스"를 참조하십시오.
      - **spec.providerCredentials**: (필수) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
        - **spec.providerCredentials.valueFromSecret**: (필수) 자격 증명 값이 비밀 키에서 가져와야 함을 나타냅니다.
          - **key**: (필수) 선택할 시크릿의 유효한 키입니다.
          - **name**: (필수) 이 필드의 값을 포함하는 시크릿의 이름입니다. 동일한 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 서비스(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure)를 지정합니다. 가능한 값:
  - aws
  - Azure
  - gcp
  - generic-s3
  - ONTAP S3
  - storagegrid-s3

c. `trident-protect-appvault-secondary-destination.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. 타겟 클러스터에서 AppMirrorRelationship CR 파일을 생성합니다.

> CR을 사용할 때는 CR을 적용하기 전에 타겟 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

CR을 사용하여 **AppMirrorRelationship**을 생성합니다

a. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-relationship.yaml`).

b. 다음 속성을 구성하십시오.

- **metadata.name:** (필수) AppMirrorRelationship 사용자 지정 리소스의 이름입니다.
- **spec.destinationAppVaultRef**: (필수) 이 값은 타겟 클러스터의 타겟 애플리케이션에 대한 AppVault 이름과 일치해야 합니다.
- **spec.namespaceMapping**: (필수) 타겟 및 소스 네임스페이스는 각 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 CR에 정의한 소스 애플리케이션의 UID와 일치해야 합니다.
- **spec.storageClassName**: (선택 사항) 클러스터에서 유효한 스토리지 클래스의 이름을 선택하십시오. 스토리지 클래스는 소스 환경과 피어링된 ONTAP 스토리지 VM에 연결되어 있어야 합니다. 스토리지 클래스를 제공하지 않으면 클러스터의 기본 스토리지 클래스가 기본적으로 사용됩니다.
- **spec.recurrenceRule**: UTC 시간으로 시작 날짜와 반복 간격을 정의합니다.

  YAML 예:

```
---
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. `trident-protect-relationship.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

CLI를 사용하여 **AppMirrorRelationship** 생성

a. AppMirrorRelationship 객체를 생성하고 적용합니다. 괄호 안의 값은 사용자 환경의 정보로 바꿔야 합니다.

```
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) 타겟 클러스터에서 복제 관계의 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

타겟 클러스터로 페일오버

Trident Protect를 사용하면 복제된 애플리케이션을 타겟 클러스터로 페일오버할 수 있습니다. 이 절차는 복제 관계를 중지하고 타겟 클러스터에서 애플리케이션을 온라인 상태로 전환합니다. Trident Protect는 소스 클러스터에서 애플리케이션이 작동 중인 경우 해당 애플리케이션을 중지하지 않습니다.

단계

1. 타겟 클러스터에서 AppMirrorRelationship CR 파일(예: `trident-protect-relationship.yaml`)을 편집하고 **spec.desiredState** 값을 `Promoted`로 변경합니다.
2. CR 파일을 저장합니다.
3. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (선택 사항) 페일오버된 애플리케이션에 필요한 보호 스케줄을 생성합니다.
5. (선택 사항) 복제 관계의 상태를 확인합니다:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

페일오버된 복제 관계 재동기화

재동기화 작업은 복제 관계를 다시 설정합니다. 재동기화 작업을 수행하면 원래 소스 애플리케이션이 실행 중인 애플리케이션이 되고 타겟 클러스터에서 실행 중인 애플리케이션에 대한 변경 사항은 모두 삭제됩니다.

이 프로세스는 복제를 다시 설정하기 전에 타겟 클러스터에서 앱을 중지합니다.

페일오버 중에 타겟 애플리케이션에 기록된 모든 데이터는 손실됩니다.

단계

1. 선택 사항: 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성합니다. 이렇게 하면 소스 클러스터의 최신 변경 사항이 캡처됩니다.
2. 타겟 클러스터에서 AppMirrorRelationship CR 파일(예: `trident-protect-relationship.yaml`)을 편집하고 spec.desiredState 값을 `Established`로 변경합니다.
3. CR 파일을 저장합니다.
4. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 장애 조치된 애플리케이션을 보호하기 위해 타겟 클러스터에 보호 일정을 생성한 경우 해당 일정을 제거합니다. 남아 있는 모든 일정은 볼륨 스냅샷 실패의 원인이 됩니다.

페일오버된 복제 관계 역 재동기화

장애 조치 복제 관계를 역동기화하면 타겟 애플리케이션이 소스 애플리케이션이 되고 소스가 타겟이 됩니다. 장애 조치 중에 타겟 애플리케이션에 변경된 내용은 유지됩니다.

단계

1. 원래 타겟 클러스터에서 AppMirrorRelationship CR을 삭제합니다. 이렇게 하면 타겟이 소스가 됩니다. 새 타겟 클러스터에 보호 스케줄이 남아 있는 경우 이를 제거합니다.
2. 원래 관계를 설정하는 데 사용한 CR 파일을 반대 클러스터에 적용하여 복제 관계를 설정합니다.
3. 새 타겟(원본 소스 클러스터)에 AppVault CR이 모두 구성되어 있는지 확인합니다.
4. 반대쪽 클러스터에 복제 관계를 설정하여 반대 방향에 대한 값을 구성합니다.

애플리케이션 복제 방향 반전

복제 방향을 반대로 하면 Trident Protect는 원래 소스 스토리지 백엔드로 계속 복제하면서 애플리케이션을 타겟 스토리지 백엔드로 이동합니다. Trident Protect는 소스 애플리케이션을 중지하고 타겟 앱으로 페일오버하기 전에 데이터를 타겟으로 복제합니다.

이 상황에서는 소스와 타겟을 교체하는 것입니다.

단계

1. 소스 클러스터에서 종료 스냅샷을 생성합니다.

CR을 사용하여 종료 스냅샷 생성

a. 소스 애플리케이션에 대한 보호 정책 스케줄을 비활성화합니다.

b. ShutdownSnapshot CR 파일을 생성합니다.

i. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-shutdownsnapshot.yaml`).

ii. 다음 속성을 구성하십시오.

- **metadata.name**: (필수) 사용자 지정 리소스의 이름입니다.
- **spec.AppVaultRef**: (필수) 이 값은 소스 애플리케이션에 대한 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.ApplicationRef**: (필수) 이 값은 소스 애플리케이션 CR 파일의 metadata.name 필드와 일치해야 합니다.

YAML 예:

```
---
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. `trident-protect-shutdownsnapshot.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

CLI를 사용하여 종료 스냅샷 만들기

a. 종료 스냅샷을 만들고 괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예:

```
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 확인합니다.

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. 소스 클러스터에서 다음 명령을 사용하여 **shutdownsnapshot.status.appArchivePath** 값을 찾고 파일 경로의 마지막 부분(베이스네임이라고도 함, 마지막 슬래시 뒤의 모든 부분)을 기록합니다.

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. 다음과 같은 변경 사항을 적용하여 새 타겟 클러스터에서 새 소스 클러스터로 페일오버를 수행하십시오.

> 장애 조치 절차의 2단계에서 AppMirrorRelationship CR 파일에 `spec.promotedSnapshot` 필드를 포함하고, 그 값을 위 3단계에서 기록한 기본 이름으로 설정합니다.

5. 페일오버된 복제 관계 역 재동기화에서 역방향 재동기화 단계를 수행하십시오.
6. 새 소스 클러스터에서 보호 스케줄을 활성화합니다.

결과

역복제로 인해 다음 작업이 수행됩니다.

- 원본 소스 애플리케이션의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 원래 소스 앱의 Pod는 앱의 Kubernetes 리소스를 삭제하여 정상적으로 중지됩니다(PVC 및 PV는 그대로 유지됨).
- Pod가 종료된 후 앱 볼륨의 스냅샷이 생성되어 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨을 읽기/쓰기에 사용할 수 있습니다.
- 앱의 Kubernetes 리소스는 원래 소스 앱이 종료된 후 복제된 볼륨 데이터를 사용하여 종료 전 스냅샷에서 복원됩니다.
- 복제는 역방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백

Trident Protect를 사용하면 다음 일련의 작업을 통해 페일오버 작업 후 "페일 백"을 구현할 수 있습니다. 원래 복제 방향을 복원하는 이 워크플로에서 Trident Protect는 복제 방향을 반전하기 전에 모든 애플리케이션 변경 사항을 원래 소스 애플리케이션으로 복제(재동기화)합니다.

이 프로세스는 타겟으로 페일오버가 완료된 관계에서 시작되며 다음 단계를 포함합니다.

- 장애 조치 상태로 시작합니다.
- 복제 관계를 역방향으로 다시 동기화합니다.

> 일반적인 재동기화 작업을 수행하지 마십시오. 이 작업을 수행하면 장애 조치 절차 중에 타겟 클러스터에 기록된 데이터가 손실됩니다.

- 복제 방향을 반대로 합니다.

단계

1. 페일오버된 복제 관계 역 재동기화 단계를 수행하십시오.
2. 애플리케이션 복제 방향 반전 단계를 수행하십시오.

복제 관계 삭제

언제든지 복제 관계를 삭제할 수 있습니다. 애플리케이션 복제 관계를 삭제하면 서로 아무런 관계가 없는 두 개의 별개의 애플리케이션이 생성됩니다.

단계

1. 현재 타겟 클러스터에서 AppMirrorRelationship CR을 삭제합니다.

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Trident Protect를 사용하여 애플리케이션을 마이그레이션하세요.

백업 데이터를 복원하여 클러스터 간에 또는 다른 스토리지 클래스로 애플리케이션을 마이그레이션할 수 있습니다.

애플리케이션을 마이그레이션할 때 애플리케이션에 대해 구성된 모든 실행 후크가 애플리케이션과 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 클론 생성

애플리케이션을 동일한 클러스터에 복제하려면 스냅샷 또는 백업을 생성하고 데이터를 동일한 클러스터로 복원합니다.

단계

1. 다음 중 하나를 수행합니다.
   a. "스냅샷 생성".
   b. "백업 생성".
2. 동일한 클러스터에서 스냅샷을 생성했는지 백업을 생성했는지에 따라 다음 중 하나를 수행하십시오.
   a. "스냅샷에서 데이터 복원".
   b. "백업에서 데이터 복원".

다른 클러스터로 클론

애플리케이션을 다른 클러스터로 복제(클러스터 간 복제)하려면 소스 클러스터에서 백업을 생성한 다음 해당 백업을 다른 클러스터로 복원하십시오. Trident Protect가 타겟 클러스터에 설치되어 있는지 확인하십시오.

"SnapMirror 복제"을 사용하면 서로 다른 클러스터 간에 애플리케이션을 복제할 수 있습니다.

단계

1. "백업 생성".
2. 백업이 포함된 오브젝트 스토리지 버킷에 대한 AppVault CR이 타겟 클러스터에 구성되어 있는지 확인합니다.
3. 타겟 클러스터에서 "백업에서 데이터를 복원합니다".

한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션 마이그레이션

백업을 타겟 스토리지 클래스로 복원하면 애플리케이션을 한 스토리지 클래스에서 다른 스토리지 클래스로 마이그레이션할 수 있습니다.

예를 들면(복원 CR의 시크릿 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 생성하고 이름을 지정합니다 `trident-protect-snapshot-restore-cr.yaml`.
2. 생성한 파일에서 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.appArchivePath**: 스냅샷 콘텐츠가 저장되는 AppVault 내부의 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 콘텐츠가 저장된 AppVault의 이름입니다.
   - **spec.namespaceMapping**: 복원 작업의 소스 네임스페이스를 타겟 네임스페이스로 매핑합니다. `my-source-namespace` 및 `my-destination-namespace`를 사용자 환경의 정보로 교체하십시오.

     ```
     ---
     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, 애플리케이션의 특정 리소스만 복원해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가할 수 있습니다.
   - **resourceFilter.resourceSelectionCriteria**: (필터링에 필수) `include or exclude`를 사용하여 resourceMatchers에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 resourceMatchers 매개변수를 추가하십시오.
     - **resourceFilter.resourceMatchers**: resourceMatcher 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우, 요소들은 OR 연산으로 일치하며, 각 요소 내부의 필드(그룹, 종류, 버전)는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**: (선택 사항) 필터링할 리소스의 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) "Kubernetes 문서"에 정의된 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. 예: `"trident.netapp.io/os=linux"`

  예를 들면 다음과 같습니다.

```
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. `trident-protect-snapshot-restore-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI를 사용하여 스냅샷 복원

단계

1. 스냅샷을 다른 네임스페이스로 복원하고 괄호 안의 값을 사용자 환경의 정보로 바꾸십시오.

   - `snapshot` 인수는 `<namespace>/<name>` 형식의 네임스페이스와 스냅샷 이름을 사용합니다.
   - `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 `source1:dest1,source2:dest2` 형식의 올바른 대상 네임스페이스에 매핑합니다.

     예를 들면 다음과 같습니다.

```
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 관리형 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 생성 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷 생성이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 이렇게 하면 애플리케이션 정합성 보장 스냅샷을 생성할 수 있습니다.

실행 후크의 유형

Trident Protect는 실행 가능 시점에 따라 다음과 같은 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 스냅샷 이후
- 사전 백업
- 백업 후
- 복원 후
- 페일오버 후

실행 순서

데이터 보호 작업이 실행될 때 실행 후크 이벤트는 다음 순서로 발생합니다.

1. 적용 가능한 사용자 지정 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 생성하고 실행할 수 있지만, 작업 전에 이러한 후크가 실행되는 순서는 보장되지 않으며 구성할 수도 없습니다.
2. 해당되는 경우 파일 시스템이 일시 중지됩니다. "Trident Protect를 사용한 파일 시스템 동결 구성에 대해 자세히 알아보십시오".
3. 데이터 보호 작업이 수행됩니다.
4. 동결된 파일 시스템은 해당되는 경우 동결이 해제됩니다.
5. 적용 가능한 사용자 지정 사후 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 생성하고 실행할 수 있지만, 작업 후 이러한 후크의 실행 순서는 보장되지 않으며 구성할 수도 없습니다.

동일한 유형(예: pre-snapshot)의 실행 후크를 여러 개 생성하는 경우 해당 후크의 실행 순서는 보장되지 않습니다. 하지만 유형이 다른 후크의 실행 순서는 보장됩니다. 예를 들어, 다음은 모든 유형의 후크가 포함된 구성의 실행 순서입니다.

1. 스냅샷 이전 후크가 실행되었습니다
2. 스냅샷 후 후크가 실행되었습니다

3. 사전 백업 후크 실행됨
4. 백업 후 후크 실행됨

앞의 순서 예는 기존 스냅샷을 사용하지 않는 백업을 실행할 때만 적용됩니다.

운영 환경에서 실행 후크 스크립트를 활성화하기 전에 항상 테스트해야 합니다. 'kubectl exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 활성화한 후에는 생성된 스냅샷과 백업이 일관성을 유지하는지 테스트해야 합니다. 이를 위해 앱을 임시 네임스페이스에 복제하고 스냅샷 또는 백업을 복원한 다음 앱을 테스트하면 됩니다.

스냅샷 실행 전 후크가 Kubernetes 리소스를 추가, 변경 또는 제거하는 경우 이러한 변경 사항은 스냅샷 또는 백업 및 이후의 모든 복원 작업에 포함됩니다.

사용자 지정 실행 후크에 대한 중요 참고 사항

앱 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- 실행 후크는 작업을 수행하기 위해 스크립트를 사용해야 합니다. 여러 실행 후크가 동일한 스크립트를 참조할 수 있습니다.
- Trident Protect에서는 실행 후크가 사용하는 스크립트를 실행 가능한 셸 스크립트 형식으로 작성해야 합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Trident Protect는 실행 후크 설정과 일치하는 기준을 사용하여 스냅샷, 백업 또는 복원 작업에 적용할 수 있는 후크를 결정합니다.

실행 후크는 종종 실행되는 애플리케이션의 기능을 제한하거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크의 실행 시간을 최소화해야 합니다. 실행 후크가 연결된 백업 또는 스냅샷 작업을 시작한 후 취소하더라도 백업 또는 스냅샷 작업이 이미 시작된 경우 후크는 계속 실행될 수 있습니다. 즉, 백업 후 실행 후크에서 사용되는 로직은 백업이 완료되었다고 가정할 수 없습니다.

실행 후크 필터

애플리케이션에 실행 후크를 추가하거나 편집할 때, 후크가 일치할 컨테이너를 관리하기 위해 필터를 추가할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하지만 각 이미지를 다른 용도로 사용하는 애플리케이션(예: Elasticsearch)에 유용합니다. 필터를 사용하면 실행 후크가 모든 동일한 컨테이너가 아닌 일부 컨테이너에서만 실행되는 시나리오를 만들 수 있습니다. 하나의 실행 후크에 여러 필터를 생성하는 경우 논리 AND 연산자로 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

실행 후크에 추가하는 각 필터는 정규 표현식을 사용하여 클러스터의 컨테이너를 일치시킵니다. 후크가 컨테이너와 일치하면 해당 컨테이너에서 연결된 스크립트를 실행합니다. 필터에 사용되는 정규 표현식은 RE2(Regular Expression 2) 구문을 사용하며, 이 구문은 일치 목록에서 컨테이너를 제외하는 필터를 생성하는 것을 지원하지 않습니다. Trident Protect에서 실행 후크 필터의 정규 표현식에 대해 지원하는 구문에 대한 자세한 내용은 "정규식 2(RE2) 구문 지원"을 참조하십시오.

복원 또는 클론 작업 후에 실행되는 실행 후크에 네임스페이스 필터를 추가하고 복원 또는 클론 소스와 타겟이 서로 다른 네임스페이스에 있는 경우 네임스페이스 필터는 타겟 네임스페이스에만 적용됩니다.

실행 후크 예

https://github.com/NetApp/Verda["NetApp Verda GitHub 프로젝트"]를 방문하여 Apache Cassandra 및 Elasticsearch와 같은 인기 애플리케이션의 실제 실행 후크를 다운로드하세요. 또한 예제를 살펴보고 자신만의 사용자 지정 실행 후크를 구성하는 데 필요한 아이디어를 얻을 수 있습니다.

실행 후크를 생성합니다

Trident Protect를 사용하여 앱에 대한 사용자 지정 실행 후크를 생성할 수 있습니다. 실행 후크를 생성하려면 소유자, 관리자 또는 구성원 권한이 필요합니다.

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-hook.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.

   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 실행 후크를 실행할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.stage**: (필수) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 가능한 값:
     - 사전
     - 게시
   - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치하는 경우 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 백업
     - 복원
     - 페일오버
   - **spec.enabled**: (선택 사항) 이 실행 후크가 활성화되었는지 비활성화되었는지 나타냅니다. 지정하지 않으면 기본값은 true입니다.
   - **spec.hookSource**: (필수) base64로 인코딩된 훅 스크립트를 포함하는 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) 실행 후크 필터를 구성하는 각 쌍의 기준 키-값 쌍의 선택적 목록입니다. 실행 후크당 최대 10개의 필터를 추가할 수 있습니다.
   - **spec.matchingCriteria.type**: (선택 사항) 실행 후크 필터 유형을 식별하는 문자열입니다. 가능한 값:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (선택 사항) 실행 후크 필터 값을 식별하는 문자열 또는 정규 표현식입니다.

     YAML 예:

```
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 입력한 후 CR을 적용합니다.

```
kubectl apply -f trident-protect-hook.yaml
```

CLI 사용

단계

1. 실행 후크를 생성할 때 괄호 안의 값을 환경 정보로 바꾸세요. 예를 들면 다음과 같습니다.

```
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 또는 오류 발생 후 수동으로 실행 후크를 다시 실행해야 하는 경우 수동으로 실행 후크를 실행할 수 있습니다. 수동으로 실행 후크를 실행하려면 소유자, 관리자 또는 멤버 권한이 필요합니다.

실행 후크를 수동으로 실행하는 것은 기본적으로 두 단계로 구성됩니다.

1. 리소스 백업을 생성합니다. 이 백업은 리소스를 수집하고 백업을 생성하며 후크가 실행될 위치를 결정합니다
2. 백업에 대해 실행 후크를 실행합니다

1단계: 리소스 백업 생성

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-resource-backup.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 리소스 백업을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - **spec.appVaultRef**: (필수) 백업 콘텐츠가 저장되는 AppVault의 이름입니다.
   - **spec.appArchivePath**: 백업 콘텐츠가 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 예:

     ```
     ---
     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 입력한 후 CR을 적용합니다.

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

CLI 사용

단계

1. 백업을 생성할 때 괄호 안의 값을 사용자 환경 정보로 바꿔주세요. 예를 들면 다음과 같습니다.

   ```
   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 확인하세요. 이 예시 명령어를 작업이 완료될 때까지 반복해서 사용할 수 있습니다.

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. 백업이 성공했는지 확인합니다.

```
kubectl describe resourcebackup <my_backup_name>
```

2단계: 실행 후크 실행

CR 사용

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다 `trident-protect-hook-run.yaml`.
2. 다음 속성을 Trident Protect 환경 및 클러스터 구성에 맞게 구성합니다.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.applicationRef**: (필수) 이 값이 1단계에서 생성한 ResourceBackup CR의 애플리케이션 이름과 일치하는지 확인하십시오.
   - **spec.appVaultRef**: (필수) 이 값이 1단계에서 생성한 ResourceBackup CR의 appVaultRef와 일치하는지 확인하십시오.
   - **spec.appArchivePath**: 이 값이 1단계에서 생성한 ResourceBackup CR의 appArchivePath와 일치하는지 확인하세요.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (필수) 지정된 실행 후크 필터가 모두 일치하는 경우 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 백업
     - 복원
     - 페일오버
   - **spec.stage**: (필수) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 이 후크 실행은 다른 단계의 후크를 실행하지 않습니다. 가능한 값:
     - 사전
     - 게시

     YAML 예:

```
---
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 입력한 후 CR을 적용합니다.

```
kubectl apply -f trident-protect-hook-run.yaml
```

CLI 사용

단계

1. 수동 실행 후크 실행 요청을 생성합니다.

```
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. exechooksrun 객체를 설명하여 최종 세부 정보 및 상태를 확인합니다.

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Trident Protect를 제거합니다

제품의 평가판에서 정식 버전으로 업그레이드하는 경우 Trident Protect 구성 요소를 제거해야 할 수도 있습니다.

Trident Protect를 제거하려면 다음 단계를 수행하십시오.

단계

1. Trident Protect CR 파일을 제거합니다.

   이 단계는 버전 25.06 이상에서는 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 Protect 블로그

여기에서 훌륭한 NetApp Trident 및 Trident Protect 블로그를 찾을 수 있습니다:

Trident 블로그

- 2025년 10월 16일: "Kubernetes를 위한 고급 스토리지 솔루션"
- 2025년 8월 19일: "데이터 일관성 향상: Trident를 사용한 OpenShift 가상화의 볼륨 그룹 스냅샷"
- 2025년 5월 9일: "Amazon EKS 애드온을 사용한 FSx for ONTAP의 자동 Trident 백엔드 구성"
- 2025년 4월 15일: "NetApp Trident와 SMB 프로토콜용 Google Cloud NetApp Volumes"
- 2025년 4월 14일: "Kubernetes의 영구 스토리지를 위한 Trident 25.02로 Fiber Channel 프로토콜 활용"
- 2025년 4월 14일: "쿠버네티스 블록 스토리지를 위한 NetApp ASA r2 시스템의 성능 활용하기"
- 2025년 3월 31일: "새로운 인증된 Operator를 통해 Red Hat OpenShift에서 Trident 설치 간소화"
- 2025년 3월 27일: "Google Cloud NetApp Volumes를 사용한 SMB용 Trident 프로비저닝"
- 2025년 3월 5일: "원활한 iSCSI 스토리지 통합 활용: AWS용 ROSA 클러스터의 FSxN 가이드"
- 2025년 2월 27일: "Trident, GKE 및 Google Cloud NetApp Volumes를 사용하여 클라우드 ID 배포"
- 2024년 12월 12일: "Trident의 Fibre Channel 지원 소개"
- 2024년 11월 11일: "NetApp Trident와 Google Cloud NetApp Volumes"
- 2024년 10월 29일: "Trident를 사용하는 Amazon FSx for NetApp ONTAP with Red Hat OpenShift Service on AWS(ROSA)"
- 2024년 10월 29일: "ROSA 및 Amazon FSx for NetApp ONTAP에서 OpenShift Virtualization을 사용한 VM의 라이브 마이그레이션"
- 2024년 7월 8일: "NVMe/TCP를 사용하여 Amazon EKS에서 최신 컨테이너화된 애플리케이션을 위한 ONTAP 스토리지 사용"
- 2024년 7월 1일: "Google Cloud NetApp Volumes Flex 및 Astra Trident를 사용한 원활한 Kubernetes 스토리지"
- 2024년 6월 11일: "OpenShift의 통합 이미지 레지스트리를 위한 백엔드 스토리지로서의 ONTAP"

Trident Protect 블로그

- 2025년 5월 16일: "Trident Protect 사후 복원 후크를 사용하여 재해 복구를 위한 레지스트리 페일오버 자동화"
- 2025년 5월 16일: "OpenShift Virtualization 재해 복구 with NetApp Trident Protect"
- 2025년 5월 13일: "Trident Protect 백업 및 복원을 통한 스토리지 클래스 마이그레이션"
- 2025년 5월 9일: "Trident Protect 복원 후 후크를 사용하여 Kubernetes 애플리케이션 재조정"
- 2025년 4월 03일: "Trident Protect Power Up: 보호 및 재해 복구를 위한 Kubernetes 복제"
- 2025년 3월 13일: "OpenShift 가상화 VM에 대한 충돌 정합성 보장 백업 및 복원 작업"
- 2025년 3월 11일: "NetApp Trident를 사용하여 애플리케이션 데이터 보호에 GitOps 패턴 확장하기"

- 2025년 3월 3일: "Trident 25.02: 새롭고 흥미로운 기능으로 Red Hat OpenShift 경험을 한 단계 끌어올리다"
- 2025년 1월 15일: "Trident Protect 역할 기반 액세스 제어 소개"
- 2024년 11월 11일: "Kubernetes 기반 데이터 관리: Trident Protect와 함께하는 새로운 시대"

지식 및 지원

자주 묻는 질문

Trident 설치, 구성, 업그레이드 및 문제 해결에 대한 자주 묻는 질문에 대한 답변을 찾아보십시오.

일반적인 질문

Trident는 얼마나 자주 출시되나요**?**

24.02 릴리스부터 Trident는 2월, 6월, 10월에 4개월마다 릴리스됩니다.

Trident는 특정 버전의 **Kubernetes**에서 릴리스된 모든 기능을 지원합니까**?**

Trident는 일반적으로 Kubernetes의 알파 기능을 지원하지 않습니다. Trident는 Kubernetes 베타 릴리스 이후 출시되는 두 번의 Trident 릴리스에서 베타 기능을 지원할 수 있습니다.

Trident가 작동을 위해 다른 **NetApp** 제품에 의존하는 부분이 있습니까**?**

Trident는 다른 NetApp 소프트웨어 제품에 대한 종속성이 없으며 독립형 애플리케이션으로 작동합니다. 단, NetApp 백엔드 스토리지 장치가 있어야 합니다.

전체 **Trident** 구성 세부 정보를 어떻게 얻을 수 있습니까**?**

`tridentctl get` 명령을 사용하여 `Trident` 구성에 대한 자세한 정보를 확인하십시오.

Trident에서 스토리지가 프로비저닝되는 방식에 대한 메트릭을 얻을 수 있습니까**?**

예. Prometheus 엔드포인트를 사용하면 관리되는 백엔드 수, 프로비저닝된 볼륨 수, 사용된 바이트 수 등 Trident 운영에 대한 정보를 수집할 수 있습니다. 모니터링 및 분석을 위해 "Cloud Insights"를 사용할 수도 있습니다.

Trident를 **CSI** 프로비저너로 사용할 때 사용자 경험이 달라지나요**?**

아니요. 사용자 경험 및 기능 측면에서 변경 사항은 없습니다. 사용되는 프로비저너 이름은 `csi.trident.netapp.io` 입니다. 현재 및 향후 릴리스에서 제공하는 모든 새로운 기능을 사용하려면 이 Trident 설치 방법을 권장합니다.

Kubernetes 클러스터에 Trident를 설치하고 사용하십시오

Trident는 프라이빗 레지스트리에서 오프라인 설치를 지원합니까**?**

예, Trident는 오프라인으로 설치할 수 있습니다. "Trident 설치에 대해 알아보세요"을(를) 참조하십시오.

Trident를 원격으로 설치할 수 있습니까**?**

예. Trident 18.10 이상 버전은 클러스터에 대한 `kubectl` 액세스 권한이 있는 모든 컴퓨터에서 원격 설치 기능을 지원합니다. `kubectl` 액세스 권한이 확인되면(예: 원격 컴퓨터에서 `kubectl get nodes` 명령을 실행하여 확인),

설치 지침을 따르십시오.

Trident로 고가용성을 구성할 수 있습니까**?**

Trident는 하나의 인스턴스로 구성된 Kubernetes Deployment(ReplicaSet)로 설치되므로 고가용성(HA)이 내장되어 있습니다. Deployment의 복제본 수를 늘리지 않는 것이 좋습니다. Trident가 설치된 노드에 장애가 발생하거나 Pod에 접근할 수 없는 경우, Kubernetes는 자동으로 Pod를 클러스터 내의 정상적인 노드에 재배포합니다. Trident는 컨트롤 플레인 전용이므로 Trident가 재배포되더라도 현재 마운트된 Pod에는 영향을 미치지 않습니다.

Trident에 **kube-system** 네임스페이스에 대한 액세스 권한이 필요합니까**?**

Trident는 애플리케이션이 새 PVC를 요청하는 시기를 확인하기 위해 Kubernetes API Server에서 읽으므로 kube-system에 대한 액세스가 필요합니다.

Trident에서 사용하는 역할 및 권한은 무엇입니까**?**

Trident 설치 프로그램은 Kubernetes ClusterRole을 생성하며, 이 역할은 클러스터의 PersistentVolume, PersistentVolumeClaim, StorageClass 및 Secret 리소스에 대한 특정 액세스 권한을 가집니다. "tridentctl 설치 사용자 지정"를 참조하십시오.

Trident가 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성할 수 있습니까**?**

필요한 경우 Trident가 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성하고 수정할 수 있습니다. "tridentctl 설치 사용자 지정"을 참조하십시오.

두 개의 서로 다른 **Kubernetes** 클러스터에 대해 두 개의 서로 다른 **Trident** 인스턴스에서 동일한 **ONTAP** 백엔드 **SVM**을 공유할 수 있습니까**?**

권장되지는 않지만, 두 개의 Trident 인스턴스에 동일한 백엔드 SVM을 사용할 수 있습니다. 설치 중 각 인스턴스에 고유한 볼륨 이름을 지정하고/또는 `StoragePrefix` 매개변수를 `setup/backend.json` 파일에 고유하게 지정하십시오. 이는 동일한 FlexVol 볼륨이 두 인스턴스 모두에 사용되지 않도록 하기 위함입니다.

ContainerLinux(이전의 **CoreOS)**에 **Trident**를 설치할 수 있습니까**?**

Trident는 단순히 Kubernetes Pod이며 Kubernetes가 실행되는 모든 곳에 설치할 수 있습니다.

Trident를 **NetApp Cloud Volumes ONTAP**와 함께 사용할 수 있습니까**?**

네, Trident는 AWS, Google Cloud 및 Azure에서 지원됩니다.

문제 해결 및 지원

NetApp에서 **Trident**를 지원합니까**?**

Trident는 오픈 소스이며 무료로 제공되지만, NetApp 백엔드가 지원되는 경우 NetApp에서 완벽하게 지원합니다.

지원 케이스를 제기하려면 어떻게 해야 합니까**?**

지원 케이스를 제기하려면 다음 중 하나를 수행하십시오.

1. Support Account Manager에게 연락하여 티켓 접수를 위한 도움을 받으세요.

2. "NetApp 지원"에 문의하여 지원 사례를 접수하십시오.

지원 로그 번들을 어떻게 생성합니까?

`tridentctl logs -a`을 실행하여 지원 번들을 생성할 수 있습니다. 번들에 캡처된 로그 외에도 Kubernetes 측의 마운트 문제를 진단하기 위해 kubelet 로그를 캡처하십시오. kubelet 로그를 얻는 방법은 Kubernetes 설치 방식에 따라 다릅니다.

새로운 기능 추가를 요청해야 할 경우 어떻게 해야 하나요?

https://github.com/NetApp/trident["Trident Github"^]에서 이슈를 생성하고 이슈의 제목과 설명에 *RFE*를 언급하십시오.

어디에 결함을 신고해야 하나요?

https://github.com/NetApp/trident["Trident Github"^]에서 문제를 생성합니다. 문제와 관련된 모든 필요한 정보와 로그를 포함해야 합니다.

Trident에 대해 명확한 설명이 필요한 간단한 질문이 있으면 어떻게 해야 하나요? 커뮤니티나 포럼이 있나요?

질문, 문제 또는 요청 사항이 있으시면 Trident "Discord 채널" 또는 GitHub를 통해 문의하십시오.

스토리지 시스템의 비밀번호가 변경되어 **Trident**가 더 이상 작동하지 않습니다. 어떻게 복구해야 하나요?

`tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`로 백엔드의 암호를 업데이트합니다. 예제에서 `myBackend`를 백엔드 이름으로, ``/path/to_new_backend.json`를 올바른 `backend.json` 파일의 경로로 바꿉니다.

Trident에서 내 **Kubernetes** 노드를 찾을 수 없습니다. 어떻게 해결해야 하나요?

Trident가 Kubernetes 노드를 찾지 못하는 이유는 크게 두 가지입니다. Kubernetes 내부의 네트워킹 문제이거나 DNS 문제일 수 있습니다. 각 Kubernetes 노드에서 실행되는 Trident 노드 데몬셋은 Trident 컨트롤러와 통신하여 노드를 Trident에 등록할 수 있어야 합니다. Trident 설치 후 네트워킹 변경이 발생한 경우, 이 문제는 클러스터에 추가된 새 Kubernetes 노드에서만 발생합니다.

Trident 포드가 손실되면 데이터가 손실되나요?

Trident Pod가 삭제되더라도 데이터는 손실되지 않습니다. Trident 메타데이터는 CRD 객체에 저장됩니다. Trident에서 프로비저닝한 모든 PV는 정상적으로 작동합니다.

Trident 업그레이드

이전 버전에서 최신 버전으로 바로 업그레이드할 수 있습니까(몇 가지 버전 건너뛰기)?

NetApp은 Trident를 한 주요 릴리스에서 바로 다음 주요 릴리스로 업그레이드하는 것을 지원합니다. 18.xx에서 19.xx로, 19.xx에서 20.xx로 등 버전 간에 업그레이드할 수 있습니다. 운영 구축 전에 랩에서 업그레이드를 테스트해야 합니다.

Trident를 이전 릴리스로 다운그레이드할 수 있습니까?

업그레이드 후 발견된 버그, 종속성 문제 또는 업그레이드 실패나 불완전한 업그레이드에 대한 수정이 필요한 경우, "Trident 제거" 해당 버전의 특정 지침을 사용하여 이전 버전을 다시 설치해야 합니다. 이것이 이전 버전으로 다운그레이드하는 유일하게 권장되는 방법입니다.

백엔드 및 볼륨 관리

ONTAP 백엔드 정의 파일에서 Management LIF와 Data LIF를 모두 정의해야 합니까?

관리 LIF는 필수입니다. DataLIF는 다양합니다.

- ONTAP SAN: iSCSI의 경우 지정하지 마십시오. Trident는 "ONTAP 선택적 LUN 매핑"를 사용하여 멀티 패스 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. `dataLIF`가 명시적으로 정의되면 경고가 생성됩니다. 자세한 내용은 "ONTAP SAN 구성 옵션 및 예"를 참조하십시오.
- ONTAP NAS: NetApp에서는 `dataLIF`를 지정하는 것을 권장합니다. 지정하지 않으면 Trident가 SVM에서 dataLIF를 가져옵니다. NFS 마운트 작업에 사용할 정규화된 도메인 이름(FQDN)을 지정하여 여러 dataLIF에 걸쳐 로드 밸런싱을 수행하는 라운드 로빈 DNS를 생성할 수 있습니다. 자세한 내용은 "ONTAP NAS 구성 옵션 및 예"를 참조하십시오.

Trident에서 ONTAP 백엔드에 대해 CHAP를 구성할 수 있습니까?

예. Trident는 ONTAP 백엔드에 대해 양방향 CHAP를 지원합니다. 이를 위해서는 백엔드 구성에서 `useCHAP=true`를 설정해야 합니다.

Trident로 엑스포트 정책을 관리하려면 어떻게 해야 합니까?

Trident는 버전 20.04부터 동적으로 내보내기 정책을 생성하고 관리할 수 있습니다. 이를 통해 스토리지 관리자는 백엔드 구성에서 하나 이상의 CIDR 블록을 제공할 수 있으며, Trident는 이러한 범위에 속하는 노드 IP를 생성되는 내보내기 정책에 추가합니다. 이러한 방식으로 Trident는 지정된 CIDR 내의 IP를 가진 노드에 대한 규칙 추가 및 삭제를 자동으로 관리합니다.

관리 및 DataLIF에 IPv6 주소를 사용할 수 있습니까?

Trident는 다음에 대한 IPv6 주소 정의를 지원합니다.

- `managementLIF` 및 `dataLIF` ONTAP NAS 백엔드용.
- ONTAP SAN 백엔드의 경우 `managementLIF`입니다. ONTAP SAN 백엔드에서는 `dataLIF`를 지정할 수 없습니다.

Trident는 IPv6에서 작동하려면 `--use-ipv6` 플래그(`tridentctl` 설치용), `IPv6`(Trident 운영자용) 또는 `tridentTPv6`(Helm 설치용)를 사용하여 설치해야 합니다.

백엔드에서 **Management LIF**를 업데이트할 수 있습니까**?**

예, `tridentctl update backend` 명령을 사용하여 백엔드 관리 LIF를 업데이트할 수 있습니다.

백엔드에서 **DataLIF**를 업데이트할 수 있습니까**?**

`ontap-nas` 및 `ontap-nas-economy`에서만 `DataLIF`를 업데이트할 수 있습니다.

Kubernetes용 **Trident**에서 여러 백엔드를 생성할 수 있습니까**?**

Trident는 동일한 드라이버 또는 서로 다른 드라이버를 사용하여 여러 백엔드를 동시에 지원할 수 있습니다.

Trident는 백엔드 자격 증명을 어떻게 저장합니까**?**

Trident는 백엔드 자격 증명을 Kubernetes Secrets로 저장합니다.

Trident는 특정 백엔드를 어떻게 선택하나요**?**

백엔드 속성을 사용하여 클래스에 적합한 풀을 자동으로 선택할 수 없는 경우 `storagePools` 및 `additionalStoragePools` 매개변수를 사용하여 특정 풀 세트를 선택합니다.

Trident가 특정 백엔드에서 프로비저닝하지 않도록 하려면 어떻게 해야 합니까**?**

`excludeStoragePools` 매개변수는 `Trident`가 프로비저닝에 사용하는 풀 집합을 필터링하는 데 사용되며 일치하는 풀을 모두 제거합니다.

동일한 유형의 백엔드가 여러 개 있는 경우 **Trident**는 어떤 백엔드를 사용할지 어떻게 선택합니까**?**

동일한 유형의 백엔드가 여러 개 구성된 경우 Trident는 `StorageClass` 및 `PersistentVolumeClaim`에 있는 매개변수를 기반으로 적절한 백엔드를 선택합니다. 예를 들어 `ontap-nas` 드라이버 백엔드가 여러 개 있는 경우 `Trident`는 `StorageClass` 및 `PersistentVolumeClaim`의 매개변수를 조합하여 일치시키고 `StorageClass` 및 `PersistentVolumeClaim`에 나열된 요구 사항을 제공할 수 있는 백엔드를 찾습니다. 요청과 일치하는 백엔드가 여러 개 있는 경우 Trident는 그중 하나를 임의로 선택합니다.

Trident는 **Element/SolidFire**에서 양방향 **CHAP**를 지원합니까**?**

예.

Trident는 **ONTAP** 볼륨에 **Qtree**를 어떻게 배포합니까**?** 단일 볼륨에 몇 개의 **Qtree**를 배포할 수 있습니까**?**

`ontap-nas-economy` 드라이버는 동일한 `FlexVol` 볼륨에 최대 `200`개의 `Qtree`(`50~300`개 사이에서 구성 가능), 클러스터 노드당 `100,000`개의 `Qtree`, 클러스터당 `240`만 개의 `Qtree`를 생성합니다. 이코노미 드라이버에서 서비스하는 새 `PersistentVolumeClaim`를 입력하면 드라이버는 새 `Qtree`를 처리할 수 있는 `FlexVol` 볼륨이 이미 존재하는지 확인합니다. `Qtree`를 처리할 수 있는 `FlexVol` 볼륨이 없는 경우 새 `FlexVol` 볼륨이 생성됩니다.

ONTAP NAS에 프로비저닝된 볼륨에 대한 **Unix** 권한을 설정하려면 어떻게 해야 합니까**?**

백엔드 정의 파일에서 매개변수를 설정하여 Trident에서 프로비저닝한 볼륨에 대한 Unix 권한을 설정할 수 있습니다.

볼륨을 프로비저닝하는 동안 **ONTAP NFS** 마운트 옵션의 명시적 세트를 구성하려면 어떻게 해야 합니까**?**

기본적으로 Trident는 Kubernetes에서 마운트 옵션을 아무 값으로도 설정하지 않습니다. Kubernetes 스토리지 클래스에서 마운트 옵션을 지정하려면 "여기"에 제공된 예제를 따르세요.

프로비저닝된 볼륨을 특정 엑스포트 정책으로 설정하려면 어떻게 해야 합니까**?**

적절한 호스트가 볼륨에 액세스할 수 있도록 하려면 백엔드 정의 파일에 구성된 `exportPolicy` 매개변수를 사용하십시오.

Trident를 사용하여 **ONTAP**에서 볼륨 암호화를 설정하려면 어떻게 해야 합니까**?**

백엔드 정의 파일의 암호화 매개변수를 사용하여 Trident에서 프로비저닝한 볼륨에 암호화를 설정할 수 있습니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방식"

Trident를 통해 **ONTAP**에 대한 **QoS**를 구현하는 가장 좋은 방법은 무엇입니까**?**

`StorageClasses`를 사용하여 ONTAP에 대한 QoS를 구현합니다.

Trident를 통해 씬 또는 씩 프로비저닝을 지정하려면 어떻게 해야 합니까**?**

ONTAP 드라이버는 씬 프로비저닝 또는 씩 프로비저닝을 지원합니다. ONTAP 드라이버는 기본적으로 씬 프로비저닝을 사용합니다. 씩 프로비저닝을 사용하려면 백엔드 정의 파일 또는 `StorageClass`를 구성해야 합니다. 둘 다 구성된 경우 `StorageClass`이 우선 적용됩니다. ONTAP에 대해 다음을 구성합니다.

1. 켜짐 `StorageClass` 상태에서 `provisioningType` 속성을 두껍게 설정하십시오.
2. 백엔드 정의 파일에서 `backend spaceReserve parameter`을(를) 볼륨으로 설정하여 씩 볼륨을 활성화하십시오.

PVC를 실수로 삭제하더라도 사용 중인 볼륨이 삭제되지 않도록 하려면 어떻게 해야 합니까**?**

Kubernetes 버전 1.10부터 PVC 보호 기능이 자동으로 활성화됩니다.

Trident에서 생성한 **NFS PVC**를 확장할 수 있습니까**?**

예. Trident에서 생성한 PVC를 확장할 수 있습니다. 단, 볼륨 자동 확장은 ONTAP 기능이며 Trident에는 적용되지 않습니다.

SnapMirror 데이터 보호**(DP)** 모드 또는 오프라인 모드인 볼륨을 가져올 수 있습니까**?**

외부 볼륨이 DP 모드이거나 오프라인 상태인 경우 볼륨 가져오기가 실패합니다. 다음과 같은 오류 메시지가 표시됩니다.

```
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 할당량이 **NetApp** 클러스터로 어떻게 변환됩니까**?**

Kubernetes 스토리지 리소스 할당량은 NetApp 스토리지 용량이 충분한 한 정상적으로 작동해야 합니다. NetApp 스토리지가 용량 부족으로 인해 Kubernetes 할당량 설정을 준수할 수 없는 경우, Trident가 프로비저닝을 시도하지만 오류가 발생합니다.

Trident를 사용하여 볼륨 스냅샷을 생성할 수 있습니까**?**

예. 주문형 볼륨 스냅샷 생성 및 스냅샷에서 영구 볼륨 생성은 Trident에서 지원됩니다. 스냅샷에서 PV를 생성하려면 `VolumeSnapshotDataSource` 기능 게이트가 활성화되어 있어야 합니다.

Trident 볼륨 스냅샷을 지원하는 드라이버는 무엇입니까**?**

오늘부터 당사의 `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` 및 `azure-netapp-files` 백엔드 드라이버에 대해 온디맨드 스냅샷 지원이 제공됩니다.

ONTAP를 사용하여 **Trident**에서 프로비저닝한 볼륨의 스냅샷 백업을 수행하려면 어떻게 해야 합니까**?**

이것은 `ontap-nas`, `ontap-san` 및 `ontap-nas-flexgroup` 드라이버에서 사용할 수 있습니다. 또한 FlexVol 레벨에서 `ontap-san-economy` 드라이버에 대해 `snapshotPolicy`를 지정할 수도 있습니다.

이 기능은 `ontap-nas-economy` 드라이버에서도 사용할 수 있지만 qtree 수준 세분성이 아닌 FlexVol 볼륨 수준 세분성으로 제공됩니다. Trident에서 프로비저닝한 볼륨을 스냅샷으로 생성하려면 백엔드 매개변수 옵션 `snapshotPolicy`을 ONTAP 백엔드에 정의된 원하는 스냅샷 정책으로 설정하십시오. 스토리지 컨트롤러에서 생성한 스냅샷은 Trident에서 인식되지 않습니다.

Trident를 통해 프로비저닝된 볼륨에 대해 스냅샷 예비 공간 비율을 설정할 수 있습니까**?**

예, 백엔드 정의 파일에서 `snapshotReserve` 속성을 설정하여 Trident를 통해 스냅샷 복사본을 저장하기 위한 특정 디스크 공간 비율을 예약할 수 있습니다. 백엔드 정의 파일에서 `snapshotPolicy` 및 `snapshotReserve`를 구성한 경우 백엔드 파일에 언급된 `snapshotReserve` 비율에 따라 스냅샷 예약 비율이 설정됩니다. `snapshotReserve` 비율 번호가 언급되지 않은 경우 ONTAP는 기본적으로 스냅샷 예약 비율을 5로 설정합니다. `snapshotPolicy` 옵션이 none으로 설정된 경우 스냅샷 예약 비율은 0으로 설정됩니다.

볼륨 스냅샷 디렉토리에 직접 액세스하여 파일을 복사할 수 있습니까**?**

예, 백엔드 정의 파일에서 `snapshotDir` 매개변수를 설정하면 Trident로 프로비저닝된 볼륨의 스냅샷 디렉터리에 액세스할 수 있습니다.

Trident를 통해 볼륨에 대한 **SnapMirror**를 설정할 수 있습니까**?**

현재 SnapMirror는 ONTAP CLI 또는 OnCommand System Manager를 사용하여 외부에서 설정해야 합니다.

영구 볼륨을 특정 **ONTAP** 스냅샷으로 복원하려면 어떻게 해야 합니까**?**

볼륨을 ONTAP 스냅샷으로 복원하려면 다음 단계를 수행하십시오.

1. 영구 볼륨을 사용 중인 애플리케이션 Pod를 중지합니다.
2. ONTAP CLI 또는 OnCommand System Manager를 통해 필요한 스냅샷으로 복원하십시오.
3. 애플리케이션 포드를 재시작합니다.

Trident는 로드 공유 미러가 구성된 **SVM**에서 볼륨을 프로비저닝할 수 있습니까**?**

NFS를 통해 데이터를 제공하는 SVM의 루트 볼륨에 대해 로드 공유 미러를 생성할 수 있습니다. ONTAP는 Trident에서 생성한 볼륨에 대한 로드 공유 미러를 자동으로 업데이트합니다. 이로 인해 볼륨 마운트가 지연될 수 있습니다. Trident를 사용하여 여러 볼륨을 생성하는 경우 볼륨 프로비저닝은 ONTAP가 로드 공유 미러를 업데이트하는 것에 따라 달라집니다.

각 고객/테넌트에 대한 스토리지 클래스 사용량을 어떻게 구분할 수 있습니까**?**

Kubernetes는 네임스페이스의 스토리지 클래스를 허용하지 않습니다. 그러나 네임스페이스별 스토리지 리소스 할당량을 사용하여 네임스페이스당 특정 스토리지 클래스의 사용을 제한할 수 있습니다. 특정 네임스페이스가 특정 스토리지에 액세스하지 못하도록 하려면 해당 스토리지 클래스의 리소스 할당량을 0으로 설정하십시오.

문제 해결

Trident를 설치하고 사용하는 동안 발생할 수 있는 문제를 해결하기 위해 여기에 제공된 포인터를 사용하십시오.

Trident에 대한 도움이 필요하면 `tridentctl logs -a -n trident`을 사용하여 지원 번들을 생성하고 NetApp 지원팀으로 보내주십시오.

일반적인 문제 해결

- Trident Pod가 제대로 시작되지 않는 경우(예: Trident Pod가 `ContainerCreating` 단계에서 준비된 컨테이너가 두 개 미만인 상태로 멈추는 경우), `kubectl -n trident describe deployment trident` 및 `kubectl -n trident describe pod trident-**`를 실행하면 추가적인 정보를 얻을 수 있습니다. kubelet 로그를 얻는 것(예: `journalctl -xeu kubelet`를 통해)도 도움이 될 수 있습니다.
- Trident 로그에 충분한 정보가 없는 경우 설치 옵션에 따라 설치 매개변수에 `-d` 플래그를 전달하여 Trident의 디버그 모드를 활성화해 볼 수 있습니다.

  그런 다음 `./tridentctl logs -n trident`을 사용하여 디버그가 설정되어 있는지 확인하고 로그에서 `level=debug msg`를 검색하십시오.

  Operator와 함께 설치됨

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  이렇게 하면 모든 Trident Pod가 다시 시작되며 몇 초 정도 걸릴 수 있습니다. `kubectl get pod -n trident`의 출력에서 'AGE' 열을 확인하여 이를 확인할 수 있습니다.

Trident 20.07 및 20.10에서는 `tprov`을(를) `torc` 대신 사용하십시오.

Helm으로 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
```

- 각 백엔드 정의에 `debugTraceFlags`를 포함하여 각 백엔드의 디버그 로그를 얻을 수도 있습니다. 예를 들어, Trident 로그에서 API 호출 및 메서드 트래버설을 얻으려면 `debugTraceFlags: {"api":true, "method":true,}`를 포함하세요. 기존 백엔드는 `debugTraceFlags`를 `tridentctl backend update`와 함께 구성할 수 있습니다.
- Red Hat Enterprise Linux CoreOS(RHCOS)를 사용하는 경우 `iscsid`이 워커 노드에서 활성화되어 기본적으로 시작되도록 해야 합니다. 이는 OpenShift MachineConfigs를 사용하거나 ignition 템플릿을 수정하여 수행할 수 있습니다.
- Trident를 "Azure NetApp Files" 사용할 때 발생할 수 있는 일반적인 문제는 테넌트 및 클라이언트 암호가 권한이 부족한 앱 등록에서 제공되는 경우입니다. Trident 요구 사항의 전체 목록은 "Azure NetApp Files" 구성을 참조하십시오.
- PV를 컨테이너에 마운트하는 데 문제가 있는 경우, `rpcbind`가 설치되어 실행 중인지 확인하십시오. 호스트 OS에 맞는 패키지 관리자를 사용하여 `rpcbind`가 실행 중인지 확인하십시오. `rpcbind` 서비스의 상태는 `systemctl status rpcbind` 또는 이에 상응하는 명령어를 실행하여 확인할 수 있습니다.
- Trident 백엔드가 이전에는 정상적으로 작동했음에도 불구하고 `failed` 상태를 보고하는 경우, 이는 백엔드와 연결된 SVM/관리자 자격 증명이 변경되었기 때문일 가능성이 높습니다. `tridentctl update backend`를 사용하여 백엔드 정보를 업데이트하거나 Trident Pod를 재시작하면 이 문제가 해결됩니다.
- Docker를 컨테이너 런타임으로 사용하여 Trident를 설치할 때 권한 문제가 발생하는 경우 `--in cluster=false` 플래그를 사용하여 Trident 설치를 시도하십시오. 이렇게 하면 설치 프로그램 Pod를 사용하지 않으므로 `trident-installer` 사용자로 인해 발생하는 권한 문제를 방지할 수 있습니다.
- 실행 실패 후 정리 작업에 `uninstall parameter <Uninstalling Trident>`를 사용하세요. 기본적으로 이 스크립트는 Trident에서 생성한 CRD를 제거하지 않으므로 실행 중인 배포 환경에서도 안전하게 제거하고 다시 설치할 수 있습니다.
- Trident의 이전 버전으로 다운그레이드하려면 먼저 `tridentctl uninstall` 명령을 실행하여 Trident를 제거하십시오. 원하는 "Trident 버전"를 다운로드하고 `tridentctl install` 명령을 사용하여 설치하십시오.
- 설치가 성공적으로 완료된 후 PVC가 `Pending` 단계에서 멈춰 있는 경우 `kubectl describe pvc`를 실행하면 Trident가 해당 PVC에 대한 PV 프로비저닝에 실패한 이유에 대한 추가 정보를 얻을 수 있습니다.

운영자를 사용한 **Trident** 배포 실패

Trident 오퍼레이터를 사용하여 Trident를 배포하는 경우 `TridentOrchestrator`의 상태가 `Installing`에서 `Installed`로 변경됩니다. `Failed` 상태를 확인했는데 오퍼레이터가 자체적으로 복구되지 않으면 다음 명령을 실행하여 오퍼레이터 로그를 확인해야 합니다.

```
tridentctl logs -l trident-operator
```

trident-operator 컨테이너의 로그를 추적하면 문제가 있는 위치를 파악할 수 있습니다. 예를 들어, 이러한 문제 중 하나는 에어갭 환경에서 업스트림 레지스트리로부터 필요한 컨테이너 이미지를 가져올 수 없는 경우일 수 있습니다.

Trident 설치가 실패한 이유를 이해하려면 `TridentOrchestrator` 상태를 확인해야 합니다.

```
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 Trident를 설치하는 데 사용된 `TridentOrchestrator`이(가) 이미 있음을 나타냅니다. 각 Kubernetes 클러스터에는 Trident 인스턴스가 하나만 있을 수 있으므로 운영자는 언제든지 생성할 수 있는 활성 `TridentOrchestrator`이(가) 하나만 존재하도록 보장합니다.

또한 Trident Pod의 상태를 관찰하면 문제가 있는지 여부를 파악하는 데 도움이 될 수 있습니다.

```
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
```

하나 이상의 컨테이너 이미지를 가져오지 못했기 때문에 Pod가 완전히 초기화되지 못하는 것을 명확하게 확인할 수 있습니다.

이 문제를 해결하려면 `TridentOrchestrator` CR을 편집해야 합니다. 또는 `TridentOrchestrator`를 삭제하고 수정된 정확한 정의로 새 CR을 생성할 수도 있습니다.

`tridentctl`을 사용한 Trident 배포 실패

무엇이 잘못되었는지 파악하는 데 도움이 되도록 `-d` 인수를 사용하여 설치 프로그램을 다시 실행할 수 있습니다. 이렇게 하면 디버그 모드가 활성화되어 문제가 무엇인지 이해하는 데 도움이 됩니다.

```
./tridentctl install -n trident -d
```

문제를 해결한 후에는 다음과 같이 설치를 정리하고 `tridentctl install` 명령을 다시 실행하십시오.

```
./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 및 CRD를 완전히 제거합니다

Trident와 생성된 모든 CRD 및 관련 사용자 지정 리소스를 완전히 제거할 수 있습니다.

> 이 작업은 되돌릴 수 없습니다. Trident를 완전히 새로 설치하려는 경우가 아니면 이 작업을 수행하지 마십시오. CRD를 제거하지 않고 Trident를 제거하려면 "Trident 제거"을(를) 참조하십시오.

Trident 운영자

Trident 운영자를 사용하여 Trident를 제거하고 CRD를 완전히 삭제하려면 다음을 수행합니다.

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Helm

Helm을 사용하여 Trident를 제거하고 CRD를 완전히 제거하려면 다음을 수행합니다.

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>tridentctl</code>

`tridentctl`을 사용하여 Trident를 제거한 후 CRD를 완전히 제거하려면

```
tridentctl obliviate crd
```

Kubernetes 1.26에서 RWX raw 블록 네임스페이스를 사용하는 NVMe 노드 언스테이징 실패

Kubernetes 1.26을 실행 중인 경우 RWX raw 블록 네임스페이스와 함께 NVMe/TCP를 사용할 때 노드 언스테이징이 실패할 수 있습니다. 다음 시나리오는 장애에 대한 해결 방법을 제공합니다. 또는 Kubernetes를 1.27로 업그레이드할 수 있습니다.

네임스페이스와 **Pod**를 삭제했습니다

Trident 관리형 네임스페이스(NVMe 영구 볼륨)가 Pod에 연결되어 있는 시나리오를 생각해 보겠습니다. ONTAP 백엔드에서 네임스페이스를 직접 삭제하면 Pod 삭제 시도 후 스테이징 해제 프로세스가 멈춥니다. 이 시나리오는 Kubernetes 클러스터 또는 다른 기능에는 영향을 미치지 않습니다.

해결 방법

해당 네임스페이스에 해당하는 영구 볼륨을 해당 노드에서 마운트 해제하고 삭제하십시오.

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
dataLIF를 가동하여 전체 기능을 복원하십시오.
```

```
 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
 `hostNQN`을 서브시스템에 다시 추가하세요.
```

NFSv4.2 클라이언트는 ONTAP 업그레이드 후 "v4.2-xattrs"가 활성화될 것으로 예상할 때 "잘못된 인수"를 보고합니다

ONTAP 업그레이드 후 NFSv4.2 클라이언트가 NFSv4.2 내보내기를 마운트하려고 할 때 "잘못된 인수" 오류를 보고할 수 있습니다. 이 문제는 SVM에서 `v4.2-xattrs` 옵션이 활성화되지 않은 경우 발생합니다. .해결 방법 SVM에서 `v4.2-xattrs` 옵션을 활성화하거나 ONTAP 9.12.1 이상으로 업그레이드하십시오. 이 옵션은 기본적으로 활성화되어 있습니다.

지원

NetApp은 다양한 방식으로 Trident를 지원합니다. 기술 자료(KB) 문서 및 Discord 채널과 같은 광범위한 무료 자가 지원 옵션을 연중무휴 24시간 이용할 수 있습니다.

Trident 지원 라이프사이클

Trident는 버전에 따라 세 가지 수준의 지원을 제공합니다. "NetApp 소프트웨어 버전 정의 지원"을 참조하십시오.

완전 지원
: Trident는 출시일로부터 12개월 동안 완벽한 지원을 제공합니다.

제한적 지원
: Trident는 출시일로부터 13개월에서 24개월 동안 제한적인 지원을 제공합니다.

자가 지원
: Trident 문서는 출시일로부터 25~36개월 동안 제공됩니다.

버전	완전 지원	제한적 지원	자가 지원
"25.10"	2026년 10월	2027년 10월	2028년 10월
"25.06"	2026년 6월	2027년 6월	2028년 6월
"25.02"	2026년 2월	2027년 2월	2028년 2월
"24.10"	—	2026년 10월	2027년 10월

"24.06"	—	2026년 6월	2027년 6월
"24.02"	—	2026년 2월	2027년 2월
"23.10"	—	—	2026년 10월
"23.07"	—	—	2026년 7월
"23.04"	—	—	2026년 4월
"23.01"	—	—	2026년 1월

자가 지원

문제 해결 관련 문서의 전체 목록을 보려면 "NetApp 기술 자료(로그인 필요)"을 참조하십시오.

커뮤니티 지원

저희 "Discord 채널"에는 컨테이너 사용자(Trident 개발자 포함)들의 활발한 공개 커뮤니티가 있습니다. 이곳은 프로젝트에 대한 일반적인 질문을 하고 비슷한 생각을 가진 동료들과 관련 주제에 대해 토론하기에 좋은 곳입니다.

NetApp 기술 지원

Trident에 대한 도움이 필요하면 `tridentctl logs -a -n trident`을 사용하여 지원 번들을 생성하고 `NetApp Support <Getting Help>`로 보내십시오.

자세한 내용은

- "Trident 리소스"
- "Kubernetes Hub"

참조

Trident 포트

Trident가 통신에 사용하는 포트에 대해 자세히 알아보십시오.

개요

Trident는 Kubernetes 클러스터 내부 및 스토리지 백엔드와의 통신을 위해 다양한 포트를 사용합니다. 다음은 주요 포트, 해당 용도 및 보안 고려 사항에 대한 요약입니다.

- 아웃바운드 포커스: Kubernetes 노드(컨트롤러 및 작업자)는 주로 스토리지 LIF/IP로 트래픽을 시작하므로 iptables 규칙은 노드 IP에서 이러한 포트의 특정 스토리지 IP로의 아웃바운드를 허용해야 합니다. 광범위한 "any-to-any" 규칙을 피하십시오.
- 인바운드 제한: 내부 Trident 포트를 클러스터 내부 트래픽으로 제한합니다(예: Calico와 같은 CNI 사용). 호스트 방화벽에서 불필요한 인바운드 노출이 없습니다.
- 프로토콜 보안:
  - 가능한 경우 TCP를 사용하십시오(더 안정적임).
  - iSCSI의 보안이 중요한 경우 CHAP/IPsec을 활성화하고, 관리에는 TLS/HTTPS(포트 443/8443)를 사용하십시오.
  - NFSv4(Trident의 기본값)의 경우, 필요하지 않다면 UDP/이전 NFSv3 포트(예: 4045-4049)를 제거하십시오.
  - 신뢰할 수 있는 서브넷으로 제한하고 Prometheus와 같은 도구를 사용하여 모니터링하십시오(선택 사항 포트 8001).

컨트롤러 노드용 포트

이 포트는 주로 Trident 운영자(백엔드 관리)를 위한 것입니다. 모든 내부 포트는 Pod 레벨이며, 호스트 방화벽이 CNI를 방해하는 경우에만 노드에서 허용합니다.

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 8000	인바운드/아웃바운드(클러스터 내부)	Trident REST 서버(operator-controller 통신)	모두	Pod CIDR로 제한하고 외부 노출을 금지합니다.
TCP 8443	인바운드/아웃바운드(클러스터 내부)	백채널 HTTPS(보안 내부 API)	모두	TLS로 암호화됨, Kubernetes 서비스 메시를 사용하는 경우 해당 서비스 메시로 제한됩니다.
TCP 8001	인바운드(클러스터 내부, 선택 사항)	Prometheus 메트릭	모두	모니터링 툴에만 노출(예: RBAC 사용), 사용하지 않을 경우 비활성화하십시오.
TCP 443	아웃바운드	ONTAP SVM/클러스터 관리 LIF에 대한 HTTPS	ONTAP(모두), ANF	TLS 인증서 유효성 검사가 필요하며, 관리 LIF IP로만 제한합니다.
TCP 8443	아웃바운드	E-Series 웹 서비스 프록시에 대한 HTTPS	E-Series(iSCSI)	기본 REST API, 인증서 사용, 백엔드 YAML에서 구성 가능.

작업자 노드용 포트

이 포트는 CSI 노드 데몬셋 및 POD 마운트용입니다. 데이터 포트는 스토리지 데이터 LIF로 아웃바운드되며, NFSv3을 사용하는 경우 NFSv3 추가 항목을 포함합니다(NFSv4의 경우 선택 사항).

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 17546	인바운드(포드에 대한 로컬)	CSI 노드 활성/준비 프로브	모두	구성 가능(--probe-port), 호스트 충돌이 없는지 확인, 로컬 전용.
TCP 8000	인바운드/아웃바운드(클러스터 내부)	Trident REST 서버	모두	위와 같이, pod 내부입니다.
TCP 8443	인바운드/아웃바운드(클러스터 내부)	백채널 HTTPS	모두	위와 같습니다.
TCP 8001	인바운드(클러스터 내부, 선택 사항)	Prometheus 메트릭	모두	위와 같습니다.
TCP 443	아웃바운드	ONTAP SVM/클러스터 관리 LIF에 대한 HTTPS	ONTAP(모두), ANF	위와 같음, 검색에 사용됩니다.
TCP 8443	아웃바운드	E-Series 웹 서비스 프록시에 대한 HTTPS	E-Series(iSCSI)	위와 같습니다.
TCP/UDP 111	아웃바운드	RPCBIND/portmapper	ONTAP-NAS(NFSv3/v4), ANF(NFS)	v3에 필요하며, v4에서는 선택 사항입니다(방화벽 오프로드). NFSv4 전용을 사용하는 경우 제한하십시오.
TCP/UDP 2049	아웃바운드	NFS 데몬	ONTAP-NAS(NFSv3/v4), ANF(NFS)	핵심 데이터, 잘 알려진 데이터, 안정성을 위해 TCP를 사용합니다.
TCP/UDP 635	아웃바운드	마운트 데몬	ONTAP-NAS(NFSv3/v4), ANF(NFS)	마운팅, 양방향 콜백 가능(필요한 경우 인바운드 임시 허용).
UDP 4045	아웃바운드	NFS 잠금 관리자(nlockmgr)	ONTAP-NAS(NFSv3)	파일 잠금, v4(pNFS 핸들)의 경우 건너뛰기, UDP 전용.
UDP 4046	아웃바운드	NFS 상태 모니터(statd)	ONTAP-NAS(NFSv3)	알림, 콜백을 위해 인바운드 임시 포트(1024-65535)가 필요할 수 있습니다.
UDP 4049	아웃바운드	NFS 할당량 데몬(rquotad)	ONTAP-NAS(NFSv3)	할당량, v4의 경우 건너뜁니다.
TCP 3260	아웃바운드	iSCSI 타겟(검색/데이터/CHAP)	ONTAP-SAN(iSCSI), E-Series(iSCSI)	잘 알려진 기능입니다. 이 포트를 통한 CHAP 인증, 보안을 위해 상호 CHAP를 활성화하십시오.

포트/프로토콜	방향	목적	드라이버/프로토콜	보안 참고 사항
TCP 445	아웃바운드	SMB/CIFS	ONTAP-NAS(SMB), ANF(SMB)	잘 알려진 사실입니다. 암호화와 함께 SMB3 사용(Trident 주석 netapp.io/smb-encryption=true).
TCP/UDP 88(선택 사항)	아웃바운드	Kerberos 인증	ONTAP(Kerb를 사용하는 NFS/SMB/iSCSI)	Kerberos를 사용하는 경우(기본값 아님), 스토리지 시스템이 아닌 AD 서버에 적용됩니다.
TCP/UDP 389(선택 사항)	아웃바운드	LDAP	ONTAP(LDAP를 사용하는 NFS/SMB)	유사, 이름 확인/인증의 경우 AD로 제한합니다.

활성/준비 상태 프로브 포트는 설치 중에 `--probe-port` 플래그를 사용하여 변경할 수 있습니다. 워커 노드에서 다른 프로세스가 이 포트를 사용하고 있지 않은지 확인하는 것이 중요합니다.

Trident REST API

"tridentctl 명령 및 옵션"는 Trident REST API와 상호 작용하는 가장 쉬운 방법이지만, 원한다면 REST 엔드포인트를 직접 사용할 수도 있습니다.

REST API를 사용하는 경우

REST API는 Kubernetes가 아닌 환경에서 Trident를 독립 실행형 바이너리로 사용하는 고급 설치에 유용합니다.

보안 강화를 위해 Trident `REST API`는 `Pod` 내에서 실행될 때 기본적으로 `localhost`로 제한됩니다. 이 동작을 변경하려면 `Pod` 구성에서 `Trident`의 `-address` 인수를 설정해야 합니다.

REST API 사용

이러한 API가 호출되는 방법을 보여주는 예제를 보려면 디버그 (`-d` 플래그를 전달하세요. 자세한 내용은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하세요.

API는 다음과 같이 작동합니다.

GET

`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.

POST

POST <trident-address>/trident/v1/<object-type>

지정된 유형의 객체를 생성합니다.

- 생성할 오브젝트에 대한 JSON 구성이 필요합니다. 각 오브젝트 유형에 대한 사양은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하세요.
- 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>

명명된 리소스를 삭제합니다.

> 백엔드 또는 스토리지 클래스와 관련된 볼륨은 계속 존재하므로 별도로 삭제해야 합니다. 자세한 내용은 "tridentctl을 사용하여 Trident를 관리합니다"을 참조하십시오.

명령줄 옵션

Trident는 Trident 오케스트레이터에 대한 여러 명령줄 옵션을 제공합니다. 이러한 옵션을 사용하여 배포를 수정할 수 있습니다.

로깅

-debug

디버깅 출력을 활성화합니다.

-loglevel <level>

로깅 수준(debug, info, warn, error, fatal)을 설정합니다. 기본값은 info입니다.

Kubernetes

-k8s_pod

이 옵션 또는 `-k8s_api_server`을 사용하여 Kubernetes 지원을 활성화합니다. 이 옵션을 설정하면 Trident가 포함된 파드의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 연결합니다. 이는 Trident가 서비스 계정을 활성화한 Kubernetes 클러스터에서 파드로 실행되는 경우에만 작동합니다.

-k8s_api_server <insecure-address:insecure-port>

이 옵션 또는 `-k8s_pod`을 사용하여 Kubernetes 지원을 활성화합니다. 지정하면 Trident는 제공된 비보안 주소와 포트를 사용하여 Kubernetes API 서버에 연결합니다. 이렇게 하면 Trident를 포드 외부에 배포할 수 있지만 API 서버에 대한 비보안 연결만 지원합니다. 안전하게 연결하려면 `-k8s_pod` 옵션을 사용하여 Trident를 포드에 배포합니다.

Docker

-volume_driver <name>

Docker 플러그인을 등록할 때 사용되는 드라이버 이름. 기본값은 `netapp`입니다.

-driver_port <port-number>

UNIX 도메인 소켓이 아닌 이 포트에서 수신 대기합니다.

-config <file>

필수; 백엔드 구성 파일에 이 경로를 지정해야 합니다.

REST

-address <ip-or-host>

Trident의 REST 서버가 수신 대기해야 하는 주소를 지정합니다. 기본값은 localhost입니다. localhost에서 수신 대기하고 Kubernetes 포드 내에서 실행할 때 REST 인터페이스는 포드 외부에서 직접 액세스할 수 없습니다. `-address ""`을(를) 사용하여 포드 IP 주소에서 REST 인터페이스에 액세스할 수 있도록 합니다.

Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 대기하고 서비스를 제공하도록 구성할 수 있습니다.

-port <port-number>

Trident의 REST 서버가 수신 대기해야 하는 포트를 지정합니다. 기본값은 8000입니다.

-rest

REST 인터페이스를 활성화합니다. 기본값은 true입니다.

Kubernetes 및 Trident 객체

REST API를 사용하여 리소스 객체를 읽고 쓰면 Kubernetes 및 Trident와 상호 작용할 수 있습니다. Kubernetes와 Trident, Trident와 스토리지, 그리고 Kubernetes와 스토리지 간의 관계를 정의하는 여러 리소스 객체가 있습니다. 이러한 객체 중 일부는 Kubernetes를 통해 관리되고, 나머지는 Trident를 통해 관리됩니다.

객체는 서로 어떻게 상호 작용합니까?

객체, 그 용도, 그리고 상호 작용 방식을 이해하는 가장 쉬운 방법은 아마도 Kubernetes 사용자의 스토리지 요청 하나를 따라가는 것일 겁니다.

1. 사용자가 관리자가 이전에 구성한 Kubernetes `StorageClass`에서 특정 크기의 새로운 `PersistentVolume`을 요청하는 `PersistentVolumeClaim`을 생성합니다.
2. Kubernetes `StorageClass`는 Trident를 프로비저너로 식별하고 요청된 클래스에 대한 볼륨을 프로비저닝하는 방법을 Trident에 알려주는 매개 변수를 포함합니다.
3. Trident는 해당 클래스에 볼륨을 프로비저닝할 수 있도록 일치하는 `StorageClass`을 식별하는 동일한 이름의 자체 `Backends` 및 `StoragePools`을 확인합니다.
4. Trident는 일치하는 백엔드에 스토리지를 프로비저닝하고 두 개의 객체를 생성합니다. 하나는 Kubernetes에 볼륨을 찾고, 마운트하고, 처리하는 방법을 알려주는 `PersistentVolume`이고, 다른 하나는 `PersistentVolume`와 실제 스토리지 간의 관계를 유지하는 Trident의 볼륨입니다.
5. Kubernetes는 `PersistentVolumeClaim`를 새 `PersistentVolume`에 바인딩합니다. `PersistentVolumeClaim`를 포함하는 Pod는 실행되는 모든 호스트에 해당 PersistentVolume을 마운트합니다.

6. 사용자는 기존 PVC의 `VolumeSnapshot`를 생성하며, Trident를 가리키는 `VolumeSnapshotClass`를 사용합니다.
7. Trident는 PVC와 연결된 볼륨을 식별하고 백엔드에 해당 볼륨의 스냅샷을 생성합니다. 또한 Kubernetes가 스냅샷을 식별하는 방법을 알려주는 `VolumeSnapshotContent`도 생성합니다.
8. 사용자는 `PersistentVolumeClaim`를 생성할 수 있으며, `VolumeSnapshot`를 소스로 사용할 수 있습니다.
9. Trident는 필요한 스냅샷을 식별하고 `PersistentVolume` 및 `Volume`를 생성하는 것과 관련된 동일한 일련의 단계를 수행합니다.

Kubernetes 객체에 대한 자세한 내용은 Kubernetes 문서의 "영구 볼륨" 섹션을 읽어보시기를 강력히 권장합니다.

Kubernetes `PersistentVolumeClaim` 객체

Kubernetes `PersistentVolumeClaim` 객체는 Kubernetes 클러스터 사용자가 스토리지에 대해 요청하는 사항입니다.

표준 사양 외에도 Trident를 사용하면 백엔드 구성에서 설정한 기본값을 재정의하려는 경우 다음과 같은 볼륨별 주석을 지정할 수 있습니다.

주석	볼륨 옵션	지원되는 드라이버
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san, ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	프로토콜	모두
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	SolidFire-SAN
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

생성된 PV에 `Delete` 회수 정책이 있는 경우 Trident는 PV가 해제될 때(즉, 사용자가 PVC를 삭제할 때) PV와 백업 볼륨을 모두 삭제합니다. 삭제 작업이 실패하면 Trident는 해당 PV를 표시하고 성공하거나 PV가 수동으로 삭제될

때까지 주기적으로 작업을 재시도합니다. PV가 `Retain` 정책을 사용하는 경우 Trident는 이를 무시하고 관리자가 Kubernetes 및 백엔드에서 정리할 것으로 간주하여 볼륨을 제거하기 전에 백업하거나 검사할 수 있도록 합니다. PV를 삭제해도 Trident가 백업 볼륨을 삭제하지 않습니다. REST API(`tridentctl`를 사용하여 제거해야 합니다.

Trident는 CSI 사양을 사용하여 볼륨 스냅샷 생성을 지원합니다. 볼륨 스냅샷을 생성하고 이를 데이터 소스로 사용하여 기존 PVC를 복제할 수 있습니다. 이렇게 하면 특정 시점의 PV 복사본을 스냅샷 형태로 Kubernetes에 제공할 수 있습니다. 그런 다음 이러한 스냅샷을 사용하여 새로운 PV를 생성할 수 있습니다. `On-Demand Volume Snapshots`작동 방식은 다음을 참조하십시오.

Trident는 `cloneFromPVC` 및 `splitOnClone` 어노테이션도 제공하여 클론을 생성합니다. 이러한 어노테이션을 사용하면 CSI 구현을 사용하지 않고도 PVC를 복제할 수 있습니다.

예를 들어: 사용자가 이미 PVC라는 이름의 `mysql`를 가지고 있다면, 주석을 사용하여 `mysqlclone`라는 새 PVC를 `trident.netapp.io/cloneFromPVC: mysql`와 같이 생성할 수 있습니다. 이 주석이 설정되면, Trident는 새로 볼륨을 프로비저닝하는 대신 mysql PVC에 해당하는 볼륨을 복제합니다.

다음 사항을 고려하십시오.

- NetApp에서는 유휴 볼륨을 복제하는 것을 권장합니다.
- PVC와 그 클론은 동일한 Kubernetes 네임스페이스에 있어야 하며 동일한 스토리지 클래스를 가져야 합니다.
- `ontap-nas` 및 `ontap-san` 드라이버를 사용할 때, PVC 주석 `trident.netapp.io/splitOnClone`을 `trident.netapp.io/cloneFromPVC`와 함께 설정하는 것이 바람직할 수 있습니다. `trident.netapp.io/splitOnClone`이 `true`로 설정되면, Trident는 복제된 볼륨을 상위 볼륨에서 분리하여, 복제된 볼륨의 수명 주기를 상위 볼륨과 완전히 분리하지만 일부 스토리지 효율성을 잃게 됩니다. `trident.netapp.io/splitOnClone`을 설정하지 않거나 `false`로 설정하면, 상위 볼륨과 복제 볼륨 간에 종속성이 생성되어 복제 볼륨이 먼저 삭제되지 않는 한 상위 볼륨을 삭제할 수 없게 되지만, 백엔드에서 공간 사용량이 줄어듭니다. 복제를 분리하는 것이 합리적인 시나리오는 비어 있는 데이터베이스 볼륨을 복제하는 경우로, 이때 볼륨과 그 복제본이 크게 달라질 것으로 예상되어 ONTAP가 제공하는 스토리지 효율성의 이점을 얻지 못하는 경우입니다.

  `sample-input` 디렉토리에는 `Trident`와 함께 사용할 `PVC` 정의의 예가 포함되어 있습니다. `Trident` 볼륨과 관련된 매개변수 및 설정에 대한 전체 설명은 를 참조하십시오.

Kubernetes `PersistentVolume` 객체

Kubernetes `PersistentVolume` 객체는 Kubernetes 클러스터에서 사용할 수 있도록 제공되는 스토리지 영역을 나타냅니다. 이 객체는 해당 객체를 사용하는 Pod와는 독립적인 수명 주기를 가집니다.

Trident는 프로비저닝하는 볼륨을 기반으로 `PersistentVolume` 객체를 생성하고 Kubernetes 클러스터에 자동으로 등록합니다. 사용자가 직접 관리할 필요가 없습니다.

Trident 기반 `StorageClass`을 참조하는 PVC를 생성하면 Trident는 해당 스토리지 클래스를 사용하여 새 볼륨을 프로비저닝하고 해당 볼륨에 대한 새 PV를 등록합니다. 프로비저닝된 볼륨과 해당 PV를 구성할 때 Trident는 다음 규칙을 따릅니다.

- Trident는 Kubernetes용 PV 이름과 스토리지 프로비저닝에 사용하는 내부 이름을 생성합니다. 두 경우 모두 해당 범위 내에서 이름이 고유하도록 보장합니다.
- 볼륨 크기는 PVC에서 요청된 크기와 최대한 일치하지만 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있습니다.

Kubernetes `StorageClass` 객체

Kubernetes `StorageClass` 객체는 `PersistentVolumeClaims`에서 이름으로 지정되어 속성 집합을 가진 스토리지를 프로비저닝합니다. 스토리지 클래스 자체는 사용할 프로비저너를 식별하고 프로비저너가 이해할 수 있는 용어로 속성 집합을 정의합니다.

이는 관리자가 생성하고 관리해야 하는 두 가지 기본 객체 중 하나입니다. 다른 하나는 Trident 백엔드 객체입니다.

Trident를 사용하는 Kubernetes `StorageClass` 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 Trident에 특화되어 있으며 Trident에 해당 클래스에 대한 볼륨을 프로비저닝하는 방법을 알려줍니다.

스토리지 클래스 매개변수는 다음과 같습니다.

속성	유형	필수	설명
속성	map[string]string	아니요	아래 속성 섹션을 참조하십시오
storagePools	map[string]StringList	아니요	내의 스토리지 풀 목록에 대한 백엔드 이름 매핑
additionalStoragePools	map[string]StringList	아니요	내부 스토리지 풀 목록에 대한 백엔드 이름 매핑
excludeStoragePools	map[string]StringList	아니요	내의 스토리지 풀 목록에 대한 백엔드 이름 매핑

스토리지 속성과 해당 속성의 가능한 값은 스토리지 풀 선택 속성과 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 속성

이러한 매개변수는 특정 유형의 볼륨을 프로비저닝하는 데 사용할 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공	요청	지원 대상:
미디어[1]	문자열	HDD, 하이브리드, SSD	풀에는 이러한 유형의 미디어가 포함되어 있습니다. 하이브리드는 둘 다를 의미합니다	지정된 미디어 유형	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	문자열	씬, 씩	풀은 이 프로비저닝 방식을 지원합니다	프로비저닝 방법이 지정되었습니다	thick: 모든 ONTAP, thin: 모든 ONTAP 및 SolidFire-SAN
backendType	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀은 이러한 유형의 백엔드에 속합니다	지정된 백엔드	모든 드라이버
스냅샷	불	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
클론	불	참, 거짓	풀은 볼륨 복제를 지원합니다	클론이 활성화된 볼륨	ontap-nas, ontap-san, solidfire-san
암호화	불	참, 거짓	스토리지 풀은 암호화된 볼륨을 지원합니다.	암호화가 설정된 볼륨	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	양의 정수	풀은 이 범위의 IOPS를 보장할 수 있습니다	볼륨에서 이러한 IOPS를 보장합니다	SolidFire-SAN

[1]: ONTAP Select 시스템에서 지원되지 않습니다

대부분의 경우 요청된 값은 프로비저닝에 직접적인 영향을 미칩니다. 예를 들어, 씩 프로비저닝을 요청하면 씩 프로비저닝된 볼륨이 생성됩니다. 그러나 Element 스토리지 풀은 요청된 값이 아닌 제공된 최소 및 최대 IOPS를 사용하여 QoS 값을 설정합니다. 이 경우 요청된 값은 스토리지 풀을 선택하는 데만 사용됩니다.

이상적으로는 `attributes` 단독으로 사용하여 특정 클래스의 요구 사항을 충족하는 데 필요한 스토리지의 특성을 모델링할 수 있습니다. Trident는 사용자가 지정한 `attributes`의 모든 조건에 맞는 스토리지 풀을 자동으로 검색하고 선택합니다.

`attributes`을(를) 사용하여 클래스에 적합한 풀을 자동으로 선택할 수 없는 경우 `storagePools` 및 `additionalStoragePools` 매개 변수를 사용하여 풀을 더욱 세분화하거나 특정 풀 세트를 선택할 수 있습니다.

`storagePools` 매개변수를 사용하여 지정된 `attributes`와 일치하는 풀 집합을 더욱 제한할 수 있습니다. 즉, Trident는 프로비저닝을 위해 `attributes` 및 `storagePools` 매개변수로 식별된 풀의 교집합을 사용합니다. 두 매개변수를 단독으로 사용하거나 함께 사용할 수 있습니다.

`additionalStoragePools` 매개변수를 사용하면 `attributes` 및 `storagePools` 매개변수로 선택된 풀과 관계없이 Trident가 프로비저닝에 사용하는 풀 집합을 확장할 수 있습니다.

`excludeStoragePools` 매개변수를 사용하여 Trident가 프로비저닝에 사용하는 스토리지 풀 집합을 필터링할 수 있습니다. 이 매개변수를 사용하면 일치하는 모든 스토리지 풀이 제거됩니다.

`storagePools` 및 `additionalStoragePools` 매개변수에서 각 항목은 `<backend>:<storagePoolList>` 형식을 취하며, 여기서 `<storagePoolList>`는 지정된 백엔드에 대한 스토리지 풀의 쉼표로 구분된 목록입니다. 예를 들어, `additionalStoragePools`의 값은 `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`과 같을 수 있습니다. 이러한 목록은 백엔드와 목록 값 모두에 대해 정규 표현식 값을 허용합니다. `tridentctl get backend`을 사용하여 백엔드 및 해당 풀 목록을 가져올 수 있습니다.

Kubernetes 속성

이러한 속성은 동적 프로비저닝 중에 Trident가 스토리지 풀/백엔드를 선택하는 데 영향을 미치지 않습니다. 대신, 이러한 속성은 Kubernetes 영구 볼륨에서 지원하는 매개변수를 제공할 뿐입니다. 워커 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
fsType	문자열	ext4, ext3, xfs	블록 볼륨의 파일 시스템 유형	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	모두

속성	유형	값	설명	관련 드라이버	**Kubernetes** 버전
allowVolumeExpansion	boolean	참, 거짓	PVC 크기 증가 지원 활성화 또는 비활성화	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	문자열	즉시, WaitForFirstConsumer	볼륨 바인딩 및 동적 프로비저닝이 발생하는 시점을 선택하십시오	모두	1.19 - 1.26

- `fsType` 매개변수는 SAN LUN에 사용할 파일 시스템 유형을 제어하는 데 사용됩니다. 또한, Kubernetes는 스토리지 클래스에 `fsType`가 존재하는지 여부를 통해 파일 시스템이 존재함을 나타냅니다. 볼륨 소유권은 `fsGroup` 보안 컨텍스트를 사용하여 제어할 수 있으며, 이는 `fsType`가 설정된 경우에만 가능합니다. "Kubernetes: Pod 또는 Container에 대한 Security Context 구성"를 참조하여 `fsGroup` 컨텍스트를 사용한 볼륨 소유권 설정에 대한 개요를 확인하십시오. Kubernetes는 다음과 같은 경우에만 `fsGroup` 값을 적용합니다.

  - `fsType`이(가) 스토리지 클래스에 설정됩니다.
  - PVC 액세스 모드는 RWO입니다.

  NFS 스토리지 드라이버의 경우 파일 시스템은 이미 NFS 내보내기의 일부로 존재합니다. `fsGroup`을 사용하려면 스토리지 클래스에서 여전히 `fsType`을 지정해야 합니다. `nfs` 또는 null이 아닌 값으로 설정할 수 있습니다.

- 볼륨 확장에 대한 자세한 내용은 "볼륨 확장"를 참조하십시오.
- Trident 설치 프로그램 번들은 `sample-input/storage-class-*.yaml`에서 Trident와 함께 사용할 수 있는 여러 스토리지 클래스 정의 예제를 제공합니다. Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래스도 함께 삭제됩니다.

Kubernetes `VolumeSnapshotClass` 객체

Kubernetes `VolumeSnapshotClass` 객체는 `StorageClasses`와 유사합니다. 이러한 객체는 여러 스토리지 클래스를 정의하는 데 도움이 되며 볼륨 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 스냅샷은 하나의 볼륨 스냅샷 클래스와 연결됩니다.

스냅샷을 생성하려면 관리자가 `VolumeSnapshotClass`를 정의해야 합니다. 볼륨 스냅샷 클래스는 다음 정의를 사용하여 생성됩니다.

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

`driver`는 `csi-snapclass` 클래스의 볼륨 스냅샷 요청이 Trident에 의해 처리되도록 Kubernetes에 지정합니다. `deletionPolicy`는 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. `deletionPolicy`가 `Delete`로 설정되면 스냅샷이 삭제될 때 볼륨 스냅샷 객체와 스토리지 클러스터의 기본 스냅샷이 모두 제거됩니다. 또는 `Retain`로 설정하면 `VolumeSnapshotContent`와 물리적 스냅샷이 유지됩니다.

Kubernetes VolumeSnapshot 객체

Kubernetes VolumeSnapshot 객체는 볼륨의 스냅샷 생성을 요청하는 것입니다. PVC가 사용자가 볼륨을 요청하는 것을 나타내는 것처럼, 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷 생성을 요청하는 것입니다.

볼륨 스냅샷 요청이 들어오면 Trident는 백엔드에서 해당 볼륨의 스냅샷 생성을 자동으로 관리하고 고유한 VolumeSnapshotContent 객체를 생성하여 스냅샷을 노출합니다. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

VolumeSnapshot의 수명 주기는 소스 PVC와 무관합니다. 스냅샷은 소스 PVC가 삭제된 후에도 유지됩니다. 연결된 스냅샷이 있는 PVC를 삭제할 때 Trident는 해당 PVC의 백업 볼륨을 삭제 중 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 해당 볼륨이 제거됩니다.

Kubernetes VolumeSnapshotContent 객체

Kubernetes VolumeSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 생성된 스냅샷을 나타냅니다. 이는 PersistentVolume`와 유사하며 스토리지 클러스터에 프로비저닝된 스냅샷을 의미합니다. `PersistentVolumeClaim 및 PersistentVolume 객체와 마찬가지로 스냅샷이 생성되면 VolumeSnapshotContent 객체는 스냅샷 생성을 요청한 VolumeSnapshot 객체와 일대일 매핑을 유지합니다.

`VolumeSnapshotContent` 객체에는 `snapshotHandle`와 같이 스냅샷을 고유하게 식별하는 세부 정보가 포함되어 있습니다. 이 `snapshotHandle`는 PV 이름과 `VolumeSnapshotContent` 객체 이름의 고유한 조합입니다.

스냅샷 요청이 들어오면 Trident는 백엔드에서 스냅샷을 생성합니다. 스냅샷이 생성되면 Trident는 VolumeSnapshotContent 객체를 구성하여 Kubernetes API에 스냅샷을 노출합니다.

일반적으로 VolumeSnapshotContent 객체를 관리할 필요는 없습니다. 단, Trident 외부에서 생성된 "볼륨 스냅샷 가져오기"를 원하는 경우는 예외입니다.

Kubernetes VolumeGroupSnapshotClass 객체

Kubernetes VolumeGroupSnapshotClass 객체는 `VolumeSnapshotClass`와 유사합니다. 이러한 객체는 여러 스토리지 클래스를 정의하는 데 도움이 되며, 볼륨 그룹 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 그룹 스냅샷은 하나의 볼륨 그룹 스냅샷 클래스와 연결됩니다.

`VolumeGroupSnapshotClass`는 스냅샷 그룹을 생성하기 위해 관리자가 정의해야 합니다. 볼륨 그룹 스냅샷 클래스는 다음 정의를 사용하여 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
```

`driver`는 `csi-group-snap-class` 클래스의 볼륨 그룹 스냅샷 요청이 Trident에서 처리되도록 Kubernetes에 지정합니다. `deletionPolicy`는 그룹 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다. `deletionPolicy`가 `Delete`로 설정되면 스냅샷이 삭제될 때 볼륨 그룹 스냅샷 객체와 스토리지 클러스터의 기본 스냅샷이 모두 제거됩니다. 반대로 `Retain`로 설정하면 `VolumeGroupSnapshotContent`와 물리적 스냅샷이 유지됩니다.

Kubernetes VolumeGroupSnapshot 객체

Kubernetes VolumeGroupSnapshot 객체는 여러 볼륨의 스냅샷을 생성해 달라는 요청입니다. PVC가 사용자가 볼륨에 대해 요청하는 것을 나타내는 것처럼, 볼륨 그룹 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성해 달라는 요청입니다.

볼륨 그룹 스냅샷 요청이 들어오면 Trident는 백엔드에서 해당 볼륨에 대한 그룹 스냅샷 생성을 자동으로 관리하고 고유한 VolumeGroupSnapshotContent 객체를 생성하여 스냅샷을 노출합니다. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 해당 스냅샷을 DataSource로 사용할 수 있습니다.

VolumeGroupSnapshot의 수명 주기는 소스 PVC와 무관합니다. 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연결된 스냅샷이 있는 PVC를 삭제할 때 Trident는 해당 PVC의 백업 볼륨을 삭제 중 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 볼륨 그룹 스냅샷도 제거됩니다.

Kubernetes VolumeGroupSnapshotContent 객체

Kubernetes VolumeGroupSnapshotContent 객체는 이미 프로비저닝된 볼륨에서 생성된 그룹 스냅샷을 나타냅니다. 이는 PersistentVolume`와 유사하며 스토리지 클러스터에 프로비저닝된 스냅샷을 의미합니다. `PersistentVolumeClaim 및 PersistentVolume 객체와 마찬가지로 스냅샷이 생성되면

VolumeSnapshotContent 객체는 스냅샷 생성을 요청한 VolumeSnapshot 객체와 일대일 매핑을 유지합니다.

`VolumeGroupSnapshotContent` 객체에는 스토리지 시스템에 있는 `volumeGroupSnapshotHandle` 및 개별 volumeSnapshotHandles와 같이 스냅샷 그룹을 식별하는 세부 정보가 포함되어 있습니다.

스냅샷 요청이 들어오면 Trident는 백엔드에서 볼륨 그룹 스냅샷을 생성합니다. 볼륨 그룹 스냅샷이 생성되면 Trident는 VolumeGroupSnapshotContent 객체를 구성하여 Kubernetes API에 스냅샷을 노출합니다.

Kubernetes CustomResourceDefinition 객체

Kubernetes 사용자 정의 리소스는 관리자가 정의하고 유사한 객체를 그룹화하는 데 사용되는 Kubernetes API의 엔드포인트입니다. Kubernetes는 객체 모음을 저장하기 위한 사용자 정의 리소스 생성을 지원합니다. 이러한 리소스 정의는 `kubectl get crds`을 실행하여 얻을 수 있습니다.

사용자 정의 리소스 정의(CRD)와 관련 객체 메타데이터는 Kubernetes의 메타데이터 저장소에 저장됩니다. 따라서 Trident를 위한 별도의 저장소가 필요하지 않습니다.

Trident는 CustomResourceDefinition 객체를 사용하여 Trident 백엔드, Trident 스토리지 클래스, Trident 볼륨과 같은 Trident 객체의 ID를 보존합니다. 이러한 객체는 Trident에서 관리됩니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 도입합니다.

CRD는 Kubernetes 구성 요소입니다. 위에 정의된 리소스의 객체는 Trident에 의해 생성됩니다. 간단한 예로, `tridentctl`을 사용하여 백엔드를 생성하면 Kubernetes에서 사용할 수 있는 해당 `tridentbackends` CRD 객체가 생성됩니다.

Trident의 CRD에 대해 유념해야 할 몇 가지 사항은 다음과 같습니다.

- Trident가 설치되면 CRD 세트가 생성되며 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- tridentctl uninstall 명령을 사용하여 Trident를 제거하면 Trident Pod가 삭제되지만 생성된 CRD는 정리되지 않습니다. Trident를 완전히 제거하고 처음부터 다시 구성하는 방법을 이해하려면 "Trident 제거"를 참조하십시오.

Trident StorageClass 오브젝트

Trident는 프로비저너 필드에 `csi.trident.netapp.io`를 지정하는 Kubernetes `StorageClass` 오브젝트에 대해 일치하는 스토리지 클래스를 생성합니다. 스토리지 클래스 이름은 해당 Kubernetes StorageClass 오브젝트의 이름과 일치합니다.

Kubernetes에서는 Trident를 프로비저너로 사용하는 Kubernetes `StorageClass`가 등록될 때 이러한 객체가 자동으로 생성됩니다.

스토리지 클래스는 볼륨에 대한 요구 사항 집합으로 구성됩니다. Trident는 이러한 요구 사항을 각 스토리지 풀에 있는 특성과 일치시킵니다. 일치하면 해당 스토리지 풀은 해당 스토리지 클래스를 사용하여 볼륨을 프로비저닝하기 위한 유효한 대상이 됩니다.

REST API를 사용하여 스토리지 클래스 구성을 생성하면 스토리지 클래스를 직접 정의할 수 있습니다. 하지만 Kubernetes 배포의 경우, 새로운 Kubernetes StorageClass 객체를 등록할 때 스토리지 클래스 구성이 생성될 것으로 예상합니다.

Trident 백엔드 객체

백엔드는 Trident가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스는 여러 백엔드를 관리할 수 있습니다.

> 이는 사용자가 직접 생성하고 관리하는 두 가지 객체 유형 중 하나입니다. 다른 하나는 Kubernetes `StorageClass` 객체입니다.

이러한 객체를 구성하는 방법에 대한 자세한 내용은 "백엔드 구성"을 참조하십시오.

Trident `StoragePool` 오브젝트

스토리지 풀은 각 백엔드에서 프로비저닝에 사용할 수 있는 개별 위치를 나타냅니다. ONTAP의 경우 이는 SVM의 애그리게이트에 해당합니다. NetApp HCI/SolidFire의 경우 이는 관리자가 지정한 QoS 대역에 해당합니다. 각 스토리지 풀에는 성능 특성과 데이터 보호 특성을 정의하는 여러 가지 스토리지 속성이 있습니다.

여기 있는 다른 객체와 달리 스토리지 풀 후보는 항상 자동으로 검색되고 관리됩니다.

Trident `Volume` 오브젝트

볼륨은 프로비저닝의 기본 단위로, NFS 공유, iSCSI 및 FC LUN과 같은 백엔드 엔드포인트를 포함합니다. Kubernetes에서 이러한 볼륨은 `PersistentVolumes`에 직접적으로 대응합니다. 볼륨을 생성할 때는 볼륨의 프로비저닝 위치를 결정하는 스토리지 클래스와 크기를 지정해야 합니다.

> - Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident가 프로비저닝한 내용을 확인할 수 있습니다.
> - 스냅샷이 연결된 PV를 삭제하면 해당 Trident 볼륨의 상태가 *삭제 중*으로 업데이트됩니다. Trident 볼륨을 삭제하려면 해당 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨이 가져야 할 속성을 정의합니다.

속성	유형	필수	설명
버전	문자열	아니요	Trident API 버전("1")
이름	문자열	예	생성할 볼륨의 이름
storageClass	문자열	예	볼륨 프로비저닝 시 사용할 스토리지 클래스
크기	문자열	예	프로비저닝할 볼륨의 크기(바이트)
프로토콜	문자열	아니요	사용할 프로토콜 유형, "file" 또는 "block"
internalName	문자열	아니요	스토리지 시스템의 오브젝트 이름, Trident에서 생성됨
cloneSourceVolume	문자열	아니요	ontap (nas, san) & solidfire-*: 복제할 볼륨의 이름

속성	유형	필수	설명
splitOnClone	문자열	아니요	ONTAP(NAS, SAN): 클론을 상위 항목에서 분할합니다
snapshotPolicy	문자열	아니요	ontap-*: 사용할 스냅샷 정책
snapshotReserve	문자열	아니요	ontap-*: 스냅샷용으로 예약된 볼륨의 백분율
exportPolicy	문자열	아니요	ontap-nas*: 사용할 엑스포트 정책
snapshotDirectory	불	아니요	ontap-nas*: 스냅샷 디렉토리가 표시되는지 여부
unixPermissions	문자열	아니요	ontap-nas*: 초기 UNIX 권한
blockSize	문자열	아니요	SolidFire-*: 블록/섹터 크기
fileSystem	문자열	아니요	파일 시스템 유형
skipRecoveryQueue	문자열	아니요	볼륨 삭제 시 스토리지의 복구 큐를 건너뛰고 볼륨을 즉시 삭제합니다.

Trident는 볼륨을 생성할 때 `internalName`를 생성합니다. 이 과정은 두 단계로 구성됩니다. 첫째, 스토리지 접두사(기본 `trident` 또는 백엔드 구성의 접두사)를 볼륨 이름 앞에 추가하여 `<prefix>-<volume-name>` 형식의 이름을 생성합니다. 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 정제합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 대체합니다(따라서 내부 이름은 `<prefix>_<volume-name>`가 됩니다). Element 백엔드의 경우 밑줄을 하이픈으로 대체합니다.

볼륨 구성을 사용하여 REST API를 통해 볼륨을 직접 프로비저닝할 수 있지만, Kubernetes 배포 환경에서는 대부분의 사용자가 표준 Kubernetes `PersistentVolumeClaim` 방식을 사용할 것으로 예상됩니다. Trident는 프로비저닝 프로세스의 일부로 이 볼륨 객체를 자동으로 생성합니다.

Trident `Snapshot` 오브젝트

스냅샷은 볼륨의 시점 복사본으로, 새 볼륨을 프로비저닝하거나 상태를 복원하는 데 사용할 수 있습니다. Kubernetes에서 스냅샷은 `VolumeSnapshotContent` 객체에 직접적으로 대응합니다. 각 스냅샷은 볼륨과 연결되며, 이 볼륨은 스냅샷의 데이터 소스가 됩니다.

각 `Snapshot` 객체는 아래에 나열된 속성을 포함합니다.

속성	유형	필수	설명
버전	문자열	예	Trident API 버전("1")
이름	문자열	예	Trident 스냅샷 객체의 이름
internalName	문자열	예	스토리지 시스템의 Trident 스냅샷 객체 이름

속성	유형	필수	설명
volumeName	문자열	예	스냅샷이 생성되는 영구 볼륨의 이름입니다
volumeInternalName	문자열	예	스토리지 시스템의 연결된 Trident 볼륨 오브젝트 이름

Kubernetes에서는 이러한 객체가 자동으로 관리됩니다. Trident가 프로비저닝한 내용을 확인할 수 있습니다.

Kubernetes `VolumeSnapshot` 객체 요청이 생성되면 Trident는 백업 스토리지 시스템에서 스냅샷 객체를 생성하여 작동합니다. 이 스냅샷 객체의 `internalName`는 접두사 `snapshot-`와 `UID` 객체의 `VolumeSnapshot`를 결합하여 생성됩니다(예: `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` 및 `volumeInternalName`는 백업 볼륨의 세부 정보를 가져와 채워집니다.

Trident `ResourceQuota` 객체

Trident 데몬셋은 `system-node-critical` 우선순위 클래스(Kubernetes에서 사용 가능한 가장 높은 우선순위 클래스)를 사용하여 정상적인 노드 종료 중에 Trident가 볼륨을 식별하고 정리할 수 있도록 하며, 리소스 부하가 높은 클러스터에서 Trident 데몬셋 파드가 우선순위가 낮은 워크로드를 선점할 수 있도록 합니다.

이를 위해 Trident는 `ResourceQuota` 객체를 사용하여 Trident 데몬셋에서 "system-node-critical" 우선순위 클래스가 충족되도록 합니다. 배포 및 데몬셋 생성 전에 Trident는 `ResourceQuota` 객체를 찾고, 찾지 못하면 적용합니다.

기본 리소스 할당량 및 우선순위 클래스를 더 세밀하게 제어해야 하는 경우 `custom.yaml`를 생성하거나 `Helm` 차트를 사용하여 `ResourceQuota` 객체를 구성할 수 있습니다.

다음은 `ResourceQuota` 객체가 Trident 데몬셋의 우선순위를 지정하는 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 "Kubernetes: 리소스 할당량"을 참조하십시오.

설치 실패 시 정리 작업 `ResourceQuota`

`ResourceQuota` 객체가 생성된 후 설치가 실패하는 드문 경우에는 먼저 xref:{relative_path}../trident-managing-k8s/uninstall-trident.html["제거"]를 시도한 다음 다시 설치하십시오.

그래도 안 되면 수동으로 `ResourceQuota` 개체를 제거하세요.

제거 `ResourceQuota`

리소스 할당을 직접 제어하려면 다음 명령을 사용하여 Trident `ResourceQuota` 객체를 제거할 수 있습니다.

```
kubectl delete quota trident-csi -n trident
```

Pod Security Standards(PSS) 및 Security Context Constraints(SCC)

Kubernetes Pod 보안 표준(PSS) 및 Pod 보안 정책(PSP)은 권한 수준을 정의하고 Pod의 동작을 제한합니다. OpenShift 보안 컨텍스트 제약 조건(SCC)은 OpenShift Kubernetes Engine에 특화된 Pod 제한 사항을 정의합니다. 이러한 사용자 정의를 제공하기 위해 Trident는 설치 중에 특정 권한을 활성화합니다. 다음 섹션에서는 Trident에서 설정한 권한에 대해 자세히 설명합니다.

PSS는 Pod Security Policies(PSP)를 대체합니다. PSP는 Kubernetes v1.21에서 사용이 권장되지 않으며 v1.25에서 제거될 예정입니다. 자세한 내용은 "Kubernetes: 보안"를 참조하십시오.

필수 Kubernetes 보안 컨텍스트 및 관련 필드

권한	설명
권한 있음	CSI를 사용하려면 마운트 포인트가 양방향이어야 하므로 Trident 노드 Pod는 권한 있는 컨테이너를 실행해야 합니다. 자세한 내용은 "Kubernetes: 마운트 전파"을 참조하십시오.
호스트 네트워킹	iSCSI 데몬에 필요합니다. `iscsiadm`는 iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 프로세스 간 통신(IPC)을 사용하여 NFSD와 통신합니다.
호스트 PID	NFS의 경우 `rpc-statd`을(를) 시작해야 합니다. Trident은 NFS 볼륨을 마운트하기 전에 호스트 프로세스를 쿼리하여 `rpc-statd`이(가) 실행 중인지 확인합니다.

권한	설명
기능	SYS_ADMIN 기능은 권한 있는 컨테이너에 대한 기본 기능의 일부로 제공됩니다. 예를 들어 Docker는 권한 있는 컨테이너에 대해 다음과 같은 기능을 설정합니다: CapPrm: 0000003fffffffff CapEff: 0000003fffffffff
Seccomp	Seccomp 프로필은 권한 있는 컨테이너에서 항상 "제한 없음" 상태이므로 Trident에서 활성화할 수 없습니다.
SELinux	OpenShift에서 권한이 있는 컨테이너는 spc_t("Super Privileged Container") 도메인에서 실행되고, 권한이 없는 컨테이너는 container_t 도메인에서 실행됩니다. containerd`에서 `container-selinux`가 설치된 경우 모든 컨테이너가 `spc_t 도메인에서 실행되므로 SELinux가 사실상 비활성화됩니다. 따라서 Trident는 컨테이너에 `seLinuxOptions`를 추가하지 않습니다.
DAC	권한이 있는 컨테이너는 루트로 실행해야 합니다. 권한이 없는 컨테이너는 CSI에 필요한 Unix 소켓에 액세스하기 위해 루트로 실행됩니다.

Pod Security Standards(PSS)

레이블	설명	기본값
pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version	Trident Controller와 노드가 설치 네임스페이스에 포함될 수 있도록 허용합니다. 네임스페이스 레이블은 변경하지 마십시오.	enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>

네임스페이스 레이블을 변경하면 Pod가 스케줄링되지 않거나 "생성 오류: …" 또는 "경고: trident-csi-… " 오류가 발생할 수 있습니다. 이러한 문제가 발생하면 네임스페이스 레이블이 privileged 변경되었는지 확인하십시오. 변경된 경우 Trident를 다시 설치하십시오.

Pod Security Policies(PSP)

필드	설명	기본값
allowPrivilegeEscalation	권한 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowedCSIDrivers	Trident는 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있음
allowedCapabilities	권한이 없는 Trident 컨테이너는 기본 세트 이상의 기능을 필요로 하지 않으며 권한이 있는 컨테이너에는 가능한 모든 기능이 부여됩니다.	비어 있음

필드	설명	기본값
allowedFlexVolumes	Trident는 "FlexVolume 드라이버"를 사용하지 않으므로 허용되는 볼륨 목록에 포함되지 않습니다.	비어 있음
allowedHostPaths	Trident 노드 Pod는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정해도 아무런 이점이 없습니다.	비어 있음
allowedProcMountTypes	Trident는 어떤 것도 사용하지 않습니다 ProcMountTypes.	비어 있음
allowedUnsafeSysctls	Trident는 안전하지 않은 `sysctls`을 필요로 하지 않습니다.	비어 있음
defaultAddCapabilities	권한 있는 컨테이너에는 별도의 기능을 추가할 필요가 없습니다.	비어 있음
defaultAllowPrivilegeEscalation	권한 상승 허용은 각 Trident Pod에서 처리됩니다.	false
forbiddenSysctls	아니요 sysctls 허용되지 않습니다.	비어 있음
fsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
hostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 와 통신해야 합니다 nfsd	true
hostNetwork	iscsiadm은 iSCSI 데몬과 통신하기 위해 호스트 네트워크가 필요합니다.	true
hostPID	호스트 PID는 노드에서 `rpc-statd`이(가) 실행 중인지 확인하는 데 필요합니다.	true
hostPorts	Trident는 호스트 포트를 사용하지 않습니다.	비어 있음
privileged	Trident 노드 Pod는 볼륨을 마운트하기 위해 권한 있는 컨테이너를 실행해야 합니다.	true
readOnlyRootFilesystem	Trident 노드 Pod는 노드 파일 시스템에 기록해야 합니다.	false
requiredDropCapabilities	Trident 노드 Pod는 권한 있는 컨테이너를 실행하므로 기능을 삭제할 수 없습니다.	none
runAsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
runAsUser	Trident 컨테이너는 루트로 실행됩니다.	runAsAny
runtimeClass	Trident는 `RuntimeClasses`을 사용하지 않습니다.	비어 있음

필드	설명	기본값
seLinux	Trident는 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문에 `seLinuxOptions`을 설정하지 않습니다.	비어 있음
supplementalGroups	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
volumes	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	hostPath, projected, emptyDir

보안 컨텍스트 제약 조건(SCC)

라벨	설명	기본값
allowHostDirVolumePlugin	Trident 노드 Pod는 노드의 루트 파일 시스템을 마운트합니다.	true
allowHostIPC	NFS 볼륨을 마운트하려면 호스트 IPC가 `nfsd`와(과) 통신해야 합니다.	true
allowHostNetwork	iscsiadm은 iSCSI 데몬과 통신하기 위해 호스트 네트워크가 필요합니다.	true
allowHostPID	호스트 PID는 노드에서 `rpc-statd`이(가) 실행 중인지 확인하는 데 필요합니다.	true
allowHostPorts	Trident는 호스트 포트를 사용하지 않습니다.	false
allowPrivilegeEscalation	권한 있는 컨테이너는 권한 상승을 허용해야 합니다.	true
allowPrivilegedContainer	Trident 노드 Pod는 볼륨을 마운트하기 위해 권한 있는 컨테이너를 실행해야 합니다.	true
allowedUnsafeSysctls	Trident는 안전하지 않은 `sysctls`을 필요로 하지 않습니다.	none
allowedCapabilities	권한이 없는 Trident 컨테이너는 기본 세트 이상의 기능을 필요로 하지 않으며 권한이 있는 컨테이너에는 가능한 모든 기능이 부여됩니다.	비어 있음
defaultAddCapabilities	권한 있는 컨테이너에는 별도의 기능을 추가할 필요가 없습니다.	비어 있음
fsGroup	Trident 컨테이너는 루트로 실행됩니다.	RunAsAny
groups	이 SCC는 Trident에 특정되며 해당 사용자에게 귀속됩니다.	비어 있음

라벨	설명	기본값
`readOnlyRootFilesystem`	Trident 노드 Pod는 노드 파일 시스템에 기록해야 합니다.	`false`
`requiredDropCapabilities`	Trident 노드 Pod는 권한 있는 컨테이너를 실행하므로 기능을 삭제할 수 없습니다.	`none`
`runAsUser`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`seLinuxContext`	Trident는 현재 컨테이너 런타임과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있기 때문에 `seLinuxOptions`을 설정하지 않습니다.	비어 있음
`seccompProfiles`	권한이 있는 컨테이너는 항상 "Unconfined"로 실행됩니다.	비어 있음
`supplementalGroups`	Trident 컨테이너는 루트로 실행됩니다.	`RunAsAny`
`users`	이 SCC를 Trident 네임스페이스의 Trident 사용자에게 바인딩하기 위한 항목이 하나 제공됩니다.	해당 없음
`volumes`	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	`hostPath, downwardAPI, projected, emptyDir`

법적 고지

법적 고지사항은 저작권 표시, 상표, 특허 등에 대한 접근을 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

NetApp이 소유한 특허의 현재 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/"

오픈 소스

NetApp 소프트웨어에서 Trident에 사용된 타사 저작권 및 라이센스는 https://github.com/NetApp/trident/의 각 릴리스에 대한 공지 파일에서 확인할 수 있습니다.