Trident 26.10 문서

Trident

NetApp
November 18, 2026

This PDF was generated from https://docs.netapp.com/ko-kr/trident/index.html on November 18, 2026.
Always check docs.netapp.com for the latest.

목차

Trident 26.10 문서 1
릴리스 정보 2
새로운 기능 2
26.10.1의 새로운 기능 2
26.10의 새로운 기능 2
26.06.2의 새로운 기능 6
26.06.1의 새로운 기능 6
26.06의 새로운 기능 7
26.02.1의 새로운 기능 10
26.02의 새로운 기능 11
25.10의 새로운 기능 13
25.06.2의 변경 사항 15
25.06.1의 변경 사항 15
25.06의 변경 사항 15
25.02.1의 변경 사항 18
25.02의 변경 사항 18
24.10.1의 변경 사항 19
24.10의 변경 사항 20
24.06의 변경 사항 21
24.02의 변경 사항 22
23.10의 변경 사항 23
23.07.1의 변경 사항 23
23.07의 변경 사항 23
23.04의 변경 사항 24
23.01.1의 변경 사항 26
23.01의 변경 사항 26
22.10의 변경 사항 27
22.07 변경 28
22.04의 변경 29
22.01.1의 변경 사항 29
22.01.0의 변경 사항 29
21.10.1 변경 30
21.10.0 변경 30
알려진 문제 31
자세한 내용을 확인하십시오 32
이전 버전의 문서 32
NetApp Trident ONTAP ASA r2 스토리지 시스템 지원 33
지원되는 작업 33
지원되지 않는 작업 33

알려진 문제 34
Trident 22.10 또는 이전 버전에서 업그레이드한 후에도 볼륨은 연결된 상태로 유지됩니다. 34
VolumeSnapshots이 ReadyToUse 상태에 도달하지 못했습니다 34
대용량 파일의 임시 백업을 복원하지 못할 수 있습니다 34
시작하십시오 35
Trident에 대해 자세히 알아보십시오 35
Trident에 대해 자세히 알아보십시오 35
Trident 아키텍처 36
개념 39
Trident의 빠른 시작 43
다음 단계 44
Trident 요구사항 44
Trident에 대한 중요 정보입니다 44
지원되는 프런트엔드(오케스트레이터) 44
지원되는 백엔드(스토리지) 45
KubeVirt 및 OpenShift 가상화를 위한 Trident 지원 45
Kubernetes 및 OpenShift 호스팅 컨트롤 플레인의 Kubernetes에 대한 Trident 지원 46
피처 요구 사항 46
호스트 운영 체제를 테스트했습니다 47
호스트 구성 47
스토리지 시스템 구성 47
Trident 포트 48
컨테이너 이미지 및 해당 Kubernetes 버전 48
Trident를 설치합니다 49
Trident 설치에 대해 알아봅니다 49
Trident 26.10에 대한 중요 정보 49
시작하기 전에 49
설치 방법을 선택합니다 49
설치 모드를 선택합니다 51
방법 및 모드에 따라 프로세스를 선택합니다 52
설치 방법 간 이동 52
기타 알려진 구성 옵션 52
Trident 연산자를 사용하여 설치합니다 52
Trident 연산자 수동 배포(표준 모드) 52
Trident 연산자 수동 배포(오프라인 모드) 57
H제어(표준 모드)를 사용하여 Trident 연산자 배포 63
H제어(오프라인 모드)를 사용하여 Trident 연산자 배포 71
Trident 운영자 설치를 사용자 지정합니다 79
tridentctl을 사용하여 설치합니다 89
tridentctl을 사용하여 Trident를 설치합니다 89
Trident tridentctl 설치 사용자 정의 93

OpenShift 인증 운영자를 사용하여 설치합니다 . . . 94
OpenShift OperatorHub를 사용하여 Trident를 설치합니다 . . . 94
OpenShift 인증 Trident 운영자로 전환하세요 . . . 97
Trident를 사용합니다 . . . 101
Trident 워커 노드를 준비합니다 . . . 101
올바른 도구 선택 . . . 101
노드 서비스 검색 . . . 101
NFS 볼륨 . . . 102
iSCSI 볼륨 . . . 102
NVMe/TCP 볼륨 . . . 106
FC 볼륨을 통한 SCSI . . . 107
SMB 볼륨 프로비저닝을 위한 준비 . . . 108
COSI를 사용하여 오브젝트 스토리지 프로비저닝 . . . 110
개요 . . . 110
요구 사항 및 제한 사항 . . . 110
COSI 활성화 . . . 111
StorageGRID 오브젝트 저장소를 구성합니다. . . . 111
ONTAP S3 오브젝트 저장소 구성 . . . 112
BucketClass를 생성하고 버킷을 요청합니다 . . . 113
버킷에 워크로드 액세스 권한 부여 . . . 114
백엔드 구성 및 관리 . . . 115
Trident 백엔드 구성 . . . 115
Azure NetApp Files . . . 115
Google Cloud NetApp 볼륨 . . . 132
NetApp HCI 또는 SolidFire 백엔드를 구성합니다 . . . 169
ONTAP SAN 드라이버 . . . 174
ONTAP NAS 드라이버 . . . 201
ONTAP 백엔드와 함께 균형 잡힌 배치 사용 . . . 233
NetApp ONTAP용 Amazon FSx . . . 236
kubectl로 백엔드를 만듭니다 . . . 269
백엔드 관리 . . . 275
스토리지 클래스를 생성하고 관리합니다 . . . 284
Trident 스토리지 클래스를 생성합니다 . . . 284
Trident 스토리지 클래스 관리 . . . 286
볼륨을 프로비저닝하고 관리합니다 . . . 288
Trident 볼륨 프로비저닝 . . . 288
볼륨 이동 . . . 291
Trident 볼륨 확장 . . . 294
RWX NVMe 하위 시스템 제한 사항 이해 . . . 304
컨트롤러 확장성 . . . 305
자동 볼륨 확장 . . . 310

ONTAP 볼륨 자동 크기 조정을 구성합니다 . . . 317
자동 증가 정책 관리 . . . 321
Trident 볼륨 가져오기 . . . 330
Trident 볼륨 이름 및 레이블 사용자 지정 . . . 341
여러 네임스페이스에서 Trident NFS 볼륨 공유 . . . 343
네임스페이스 간 Trident 볼륨 복제 . . . 347
SnapMirror를 사용하여 볼륨을 복제합니다 . . . 349
CSI 토폴로지를 사용합니다 . . . 356
Trident 스냅샷 작업 . . . 362
Trident 볼륨 그룹 스냅샷 작업 . . . 370
Trident 관리 및 모니터링 . . . 375
Trident를 업그레이드합니다 . . . 375
Trident를 업그레이드합니다 . . . 375
운영자와 함께 업그레이드하십시오 . . . 376
tridentctl을 사용하여 Trident 업그레이드 . . . 380
tridentctl을 사용하여 Trident를 관리합니다 . . . 381
명령 및 글로벌 플래그 . . . 381
명령 옵션 및 플래그 . . . 383
플러그인 지원 . . . 387
Trident 스토리지 운영 모니터링 . . . 388
개요 . . . 388
1단계: Prometheus 목표를 정의합니다 . . . 388
2단계: Prometheus ServiceMonitor를 만듭니다 . . . 388
3단계: PromQL을 사용하여 Trident 메트릭 쿼리 . . . 390
Trident AutoSupport 원격 측정 기능에 대해 알아봅니다 . . . 391
Trident 메트릭을 비활성화합니다 . . . 392
Trident를 제거합니다 . . . 392
원래 설치 방법을 확인합니다 . . . 393
Trident 운영자 설치를 제거합니다 . . . 393
를 제거합니다 `tridentctl` 설치 . . . 394
Docker를 위한 Trident . . . 395
배포를 위한 사전 요구 사항 . . . 395
요구 사항을 확인합니다 . . . 395
NVMe 툴 . . . 397
FC 도구 . . . 398
Trident를 구축합니다 . . . 399
Docker 관리형 플러그인 방법(버전 1.13/17.03 이상) . . . 399
기존 방법(버전 1.12 이하) . . . 401
Trident 업그레이드 또는 제거 . . . 404
업그레이드 . . . 404
설치 제거 . . . 405

볼륨 작업 406
볼륨을 생성합니다 406
볼륨을 제거합니다 406
볼륨의 클론을 생성합니다 407
외부에서 생성된 볼륨에 액세스합니다 408
드라이버별 볼륨 옵션 408
로그를 수집합니다 413
문제 해결을 위해 로그를 수집합니다 413
일반적인 문제 해결 팁 414
여러 Trident 인스턴스를 관리합니다 414
Docker 관리 플러그인 단계(버전 1.13/17.03 이상) 414
기존(버전 1.12 이하) 단계 415
스토리지 구성 옵션 415
글로벌 구성 옵션 415
ONTAP 구성 416
Element 소프트웨어 구성 423
알려진 문제 및 제한 사항 425
이전 버전에서 Trident Docker 볼륨 플러그인을 20.10 이상으로 업그레이드하면 해당 파일 또는 디렉터리 오류가 없는 업그레이드 오류가 발생합니다. 425
볼륨 이름은 최소 2자 이상이어야 합니다. 425
Docker Swarm에는 모든 스토리지와 드라이버 조합에서 Trident이 이를 지원하지 못하는 특정한 동작이 있습니다. 425
FlexGroup를 프로비저닝하고 있는 경우, 두 번째 FlexGroup에 프로비저닝되는 FlexGroup와 공통되는 하나 이상의 애그리게이트가 있는 경우 ONTAP는 두 번째 FlexGroup를 프로비저닝하지 않습니다. 426
모범 사례 및 권장사항 427
Trident 배포 권장 사항 427
전용 네임스페이스에 구축 427
할당량 및 범위 제한을 사용하여 스토리지 사용량을 제어할 수 있습니다 427
Trident 스토리지 구성 모범 사례 427
플랫폼 개요 427
ONTAP 및 Cloud Volumes ONTAP 모범 사례 427
SolidFire 모범 사례 432
자세한 정보는 어디서 찾을 수 있습니까? 433
Trident를 Kubernetes와 통합하십시오 433
운전자 선택 및 전개 434
스토리지 클래스 설계 436
가상 풀 설계 437
볼륨 작업입니다 438
메트릭 서비스 441
Trident 데이터 보호 및 재해 복구 442
Trident 복제 및 복구 442

SVM 복제 및 복구 443
볼륨 복제 및 복구 444
스냅샷 데이터 보호 444
Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화 444
강제 분리에 대한 세부 정보 445
자동 장애 조치에 대한 세부 정보 445
보안 450
Trident 보안 권장 사항 450
Trident Linux Unified Key Setup(LUKS) 451
RWX Linux Unified Key Setup(LUKS) 볼륨에 대한 임대가 중단된 문제를 해결합니다 457
Trident Kerberos 전송 중 암호화 459
Trident Protect로 애플리케이션을 보호하세요 467
Trident Protect에 대해 알아보세요 467
다음 단계 467
Trident Protect 설치 467
Trident 프로텍트 요구 사항 467
Trident Protect 설치 및 구성 471
Trident Protect CLI 플러그인을 설치하세요 474
Trident Protect 설치 사용자 정의 476
Trident Protect 관리 481
Trident Protect 권한 및 액세스 제어 관리 481
Trident Protect 리소스 모니터링 487
Trident Protect 지원 번들 생성 491
Trident 프로텍트 업그레이드 493
애플리케이션을 관리하고 보호합니다 495
Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다. 495
Trident Protect 스냅샷 및 백업용 AppVault 메타데이터 암호화 506
Trident Protect를 사용하여 관리를 위한 애플리케이션 정의 513
Trident Protect를 사용하여 애플리케이션 보호 517
Trident Protect에서 볼륨 그룹 스냅샷 사용 527
응용 프로그램을 복원합니다 532
NetApp SnapMirror를 사용하여 애플리케이션 복제 553
Trident Protect를 사용하여 애플리케이션 마이그레이션 569
KubeVirt 가상 머신을 Trident Protect로 보호합니다 572
Trident Protect 실행 후크 관리 584
Trident Protect 제거 591
Trident 및 Trident 프로텍트 블로그 592
Trident 블로그 592
Trident 프로텍트 블로그 592
지식 및 지원 594
자주 묻는 질문 594

일반적인 질문 . . . 594
Kubernetes 클러스터에 Trident를 설치하고 사용합니다 . . . 594
문제 해결 및 지원 . . . 595
Trident를 업그레이드합니다 . . . 596
백엔드 및 볼륨 관리 . . . 596
문제 해결 . . . 601
일반 문제 해결 . . . 601
연산자를 사용한 Trident 배포 실패 . . . 602
를 사용하여 Trident를 배포하지 못했습니다 `tridentctl` . . . 604
Trident 및 CRD를 완전히 제거합니다 . . . 604
rwx 원시 블록 네임스페이스와 관련된 NVMe 노드 스테이징 해제 실패 o Kubernetes 1.26 . . . 605
ONTAP 업그레이드 후 "v4.2-xattrs"가 활성화될 것으로 예상할 때 NFSv4.2 클라이언트가 "잘못된 인수"를 보고합니다. . . . 605
지원 . . . 605
Trident 지원 라이프사이클 . . . 606
자가 지원 . . . 606
커뮤니티 지원 . . . 606
NetApp 기술 지원 . . . 606
를 참조하십시오 . . . 607
참조하십시오 . . . 608
Trident 포트 . . . 608
개요 . . . 608
Trident REST API . . . 610
REST API 사용 시기 . . . 610
REST API 사용 . . . 610
명령줄 옵션 . . . 611
로깅 . . . 611
쿠버네티스 . . . 611
Docker 를 참조하십시오 . . . 611
휴식 . . . 612
Kubernetes 및 Trident 오브젝트 . . . 612
개체가 서로 어떻게 상호 작용합니까? . . . 612
쿠버네티스 `PersistentVolumeClaim` 오브젝트 . . . 613
쿠버네티스 `PersistentVolume` 오브젝트 . . . 614
쿠버네티스 `StorageClass` 오브젝트 . . . 615
쿠버네티스 `VolumeSnapshotClass` 오브젝트 . . . 618
쿠버네티스 `VolumeSnapshot` 오브젝트 . . . 618
쿠버네티스 `VolumeSnapshotContent` 오브젝트 . . . 619
Kubernetes `VolumeGroupSnapshotClass` 오브젝트 . . . 619
Kubernetes `VolumeGroupSnapshot` 오브젝트 . . . 619

Kubernetes VolumeGroupSnapshotContent 오브젝트 620
쿠버네티스 CustomResourceDefinition 오브젝트 620
Trident 개체 StorageClass 620
Trident 백엔드 객체 621
Trident 개체 StoragePool 621
Trident 개체 Volume 621
Trident 개체 Snapshot 622
Trident 개체입니다 ResourceQuota 623
POD 보안 표준(PSS) 및 보안 컨텍스트 제약(SCC) 624
필요한 Kubernetes 보안 컨텍스트 및 관련 필드 624
POD 보안 표준(PSS) 625
PSP(POD 보안 정책) 625
SCC(Security Context Constraints) 627
법적 고지 629
저작권 629
상표 629
특허 629
개인 정보 보호 정책 629
오픈 소스 629

Trident 26.10 문서

릴리스 정보

새로운 기능

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

> 설치 프로그램 zip 파일에 제공되는 Linux용 tridentctl 바이너리는 테스트되고 지원되는 버전입니다. zip 파일의 '/Extras' 부분에 제공된 'macos' 바이너리는 테스트되거나 지원되지 않습니다.

26.10.1의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

수정

- * Kubernetes *:
  - Trident 25.10 이하 버전에서 업그레이드한 후 시작 시 Trident 노드 Pod가 사용 중인 볼륨을 강제로 분리하여 Pod가 `volume … is not staged` 오류로 실패하는 문제를 수정했습니다.

26.10의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

> Trident 26.10.0로 업그레이드하지 마십시오. 시작하는 동안 Trident 노드 Pod가 아직 사용 중인 볼륨을 분리할 수 있습니다. 영향을 받는 Pod는 오류 `volume … is not staged`로 인해 실패할 수 있습니다. 이 문제는 Trident 26.10.1에서 수정되었습니다. 대신 Trident 26.10.1 이상으로 업그레이드하십시오.

트라이던트

향상된 기능

- * Kubernetes *:
  - StorageGRID에서 오브젝트 스토리지 버킷의 COSI 기반 동적 프로비저닝에 대한 지원이 추가되었습니다. 자세한 내용은 "COSI를 사용하여 오브젝트 스토리지 프로비저닝"을 참조하십시오.
  - 구성 변경 사항을 감지하고 이에 맞춰 NVMe 데이터 LIF를 주기적으로 새로 고치는 기능이 추가되었습니다. 자세한 내용은 "NVMe 데이터 LIF 새로 고침"를 참조하십시오.
  - ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버로 프로비저닝된 새 볼륨에 대한 ONTAP Autosize 지원이 추가되었습니다. 자세한 내용은 "ONTAP 볼륨 자동 크기 조정을 구성합니다"을(를) 참조하십시오.
  - Kubernetes 1.37 지원이 추가되었습니다. 지원되는 버전 및 필요한 이미지에 대해서는 "Trident 요구사항"을(를) 참조하십시오.
  - ONTAP 볼륨 복구 대기열을 건너뛰도록 CommVault 및 Rubrik 백업에 대한 백업 클론 PVC 인식 기능을 추가했습니다(이슈 "#1182").

- 새로운 NVMe RWX 볼륨에 대한 `maxNamespacesPerSuperSubsystem` 백엔드 옵션이 추가되었습니다. 자세한 내용은 "NVMe RWX 서브시스템 제한"을(를) 참조하십시오.
- Kubernetes 배포에서 HTTPS 컨트롤러 백채널을 대체하여 `TridentNode`spec 및 status를 통해 CRD 기반 CSI 노드 등록을 추가했습니다.
- OpenShift의 Google Cloud WIF(Workload Identity Federation)에 대해 구성 가능한 대상을 사용하여 서비스 계정 토큰을 프로젝션하기 위한 설치 관리자 지원이 추가되었습니다. 설치 및 백엔드 구성에 대한 자세한 내용은 "워크로드 ID 페더레이션을 사용하여 Google Cloud NetApp Volumes 구성"을(를) 참조하십시오.
- Prometheus에서 개별 백엔드 상태를 식별하기 위해 `trident_backend_info` 메트릭에 `backend_state` 레이블을 추가했습니다(Issue "#1088"). Trident 메트릭에 대한 자세한 내용은 "Trident 모니터링"를 참조하십시오.

- iSCSI를 통해 ONTAP-SAN 드라이버로 프로비저닝된 RWX 볼륨에 대한 LUKS 암호화 지원이 추가되었습니다. 자세한 내용은 "LUKS(Linux Unified Key Setup) 사용"을 참조하십시오.
- ONTAP-NAS-FlexGroup 드라이버 및 Docker에 대한 GA 동시성 지원이 추가되었습니다. ASA r2 동시성은 이제 기술 미리보기에서 GA로 전환되었습니다. 지원되는 드라이버 및 동작에 대한 자세한 내용은 "컨트롤러 확장성"을(를) 참조하십시오.
- iSCSI 경로를 모두 사용할 수 없는 경우 경로 제거 및 iSCSI 자체 복구를 사용한 CSI 볼륨 확장에 대한 지원이 추가되었습니다. 자세한 내용은 "볼륨 확장"을(를) 참조하십시오.
- 루트 스쿼시가 적용된 ONTAP-NAS 및 ONTAP-NAS-FlexGroup 볼륨 생성 시 ONTAP 볼륨 UNIX 그룹 ID를 설정하기 위해 `trident.netapp.io/unixGroupID` PVC 주석을 추가했습니다(이슈 "#1079").
- NVMe/TCP, NFSv4 및 iSCSI에 대한 Talos OS 지원이 추가되었습니다(이슈 "#806"). 지원되는 플랫폼 및 워커 노드 요구 사항은 "Trident 요구사항" 및 "작업자 노드를 준비합니다"을 참조하십시오.
- ONTAP REST 및 ZAPI 클라이언트 타임아웃을 백엔드별로 구성할 수 있는 선택적 `storageAPITimeout` 백엔드 매개변수를 추가했습니다.
- ONTAP REST 이름 조회에 `SQL=false`를 추가하기 위한 `useRESTSQLBypass` ONTAP 백엔드 옵션이 추가되었습니다.
- ONTAP-NAS-Economy qtree 가져오기가 기술 미리보기에서 일반 출시(GA)로 승격되었습니다. 가져오기 지침은 "Trident 볼륨 가져오기"을(를) 참조하십시오.

수정

- * Kubernetes *:
  - `enableConcurrency`이(가) `true`일 때 백엔드 업데이트 또는 컨트롤러 재시작 중에 ONTAP-NAS 내보내기 정책 규칙이 잘못 삭제되는 문제를 수정했습니다(문제 "#1166", "#1179"). 내보내기 정책 구성은 "ONTAP NAS 구성 옵션 및 예"을(를) 참조하십시오.
  - 자격 증명이나 QoS 변경 등 중요하지 않은 백엔드 업데이트로 인해 볼륨별 오펀 스캔이 트리거되어 대규모 백엔드에서 CSI 작업이 차단될 수 있는 문제를 수정했습니다. 이제 스캔은 SVM 또는 스토리지 접두사가 변경될 때만 실행됩니다.
  - CSI gRPC 요청 덤프, 시작 환경 출력, 기록된 `TridentBackendConfig` 사양 등 디버그 로그에 자격 증명이 표시되는 문제를 수정했습니다. 이제 시작 출력에는 Trident `TRIDENT_*` 변수 값만 기록되고 다른 변수는 이름으로 나열됩니다.
  - CSI 게시와 볼륨 이동 간의 경쟁 조건으로 인해 오래된 액세스 정보가 있는 볼륨이 연결될 수 있었던 문제를 수정했습니다. 볼륨 이동에 대한 자세한 내용은 "볼륨 이동"을(를) 참조하십시오.
  - 노드 등록이 완료되기 전에 `NodeGetInfo`이(가) 빈 토폴로지 레이블을 반환하여 토폴로지 인식 프로비저닝 실패를 유발할 수 있었던 문제를 수정했습니다(이슈 "#1131"). 토폴로지 인식 프로비저닝에 대해서는 "CSI 토폴로지를 사용하여 스토리지 프로비저닝"을(를) 참조하십시오.

- 다른 Pod가 연결되는 동안 공유 서브시스템이 종료될 수 있는 드문 NVMe 연결 해제 경쟁 문제를 수정했습니다.
- 동시성 환경에서 마운트한 볼륨을 클론 복제할 때 발생하는 NVMe PVC 클론 게시 실패 및 잘못된 상위 네임스페이스 매핑 문제를 수정했습니다.
- `nameTemplate`(으)로 구성된 ONTAP-SAN-Economy 백엔드의 PVC에 대한 스냅샷 생성 실패 문제를 수정했습니다. 백엔드 구성 예시는 "ONTAP SAN 구성 옵션 및 예"을(를) 참조하십시오.
- REST 이름 인덱스가 업데이트되기 전에 삭제될 경우 ONTAP REST 백엔드에서 새로 생성된 FlexVol이 고아 상태가 되는 문제를 수정했습니다. 이제 Trident는 UUID를 기준으로 볼륨을 삭제합니다.
- 백엔드 LIF에 연결할 수 없을 때 컨트롤러 시작이 멈추는 문제를 수정했습니다. 영향을 받는 백엔드가 `Failed` 표시되고 시작이 계속 진행됩니다.
- 다중 키 노드 선택기로 구성된 경우 조정 시마다 Operator 관리 컨트롤러 및 노드 Pod가 재시작되는 문제가 수정되었습니다. 노드 선택기 구성은 "Trident 운영자 설치를 사용자 지정합니다"을(를) 참조하십시오.
- TridentConfigurator에서 생성된 Azure NetApp Files `TridentBackendConfig` 리소스에 생성 후 유효성 검사에 필요한 `trident.netapp.io/configurator` 레이블이 누락되는 문제를 수정했습니다.
- Trident CSI 서비스가 IPv6 ClusterIP를 사용할 때 노드 등록이 실패하는 문제를 수정했습니다(Issue "#1164").
- 오래된 볼륨 게시가 여전히 게시 취소되는 동안 워커 노드가 삭제되었다가 다시 연결될 때 Trident 노드가 영구적으로 제거되는 문제를 수정했습니다.
- Trident의 캐시된 내보내기 정책이 최신 상태가 아닐 때 이미 연결된 노드에서 ONTAP-NAS 볼륨이 NFS 내보내기 액세스 권한을 잃는 문제를 수정했습니다. 내보내기 정책 구성에 대해서는 "ONTAP NAS 구성 옵션 및 예"를 참조하십시오.
- `tridentctl` 설치 정리 과정에서 노드 복구 템플릿 CRD가 삭제되어 강제 분리가 활성화된 상태에서 추후 설치가 불가능해질 수 있는 문제를 수정했습니다. 설치에 대한 자세한 내용은 "tridentctl을 사용하여 Trident를 설치합니다"을(를) 참조하십시오.
- 재시도 지연을 추가하여 스냅샷 삭제 호출이 ONTAP API에 과부하를 일으키는 문제를 해결했습니다. 스냅샷 작업에 대해서는 "스냅샷 관리"를 참조하십시오.
- Trident 오퍼레이터 Helm 차트의 AutoSupport 이미지 헬퍼 주석과 cloudProvider 값 따옴표 처리 문제를 수정했습니다.
- CSI 재시도 시 FlexVols가 조기에 삭제되거나 생성/삭제 루프가 발생하지 않도록 진행 중이거나 기존 작업에 대한 ONTAP ZAPI 볼륨 생성 처리 방식을 수정했습니다.
- 더 이상 존재하지 않는 일관성 그룹에 대해 볼륨 그룹 스냅샷을 재시도할 때 발생하는 패닉을 수정했습니다. 볼륨 그룹 스냅샷에 대해서는 "볼륨 그룹 스냅샷 생성"을(를) 참조하십시오.
- 지원되는 최소 및 최대 Kubernetes 버전에 따른 Trident 지원 경고를 제거했습니다.

- 공유 명명 잠금에서 발생할 수 있는 경쟁 조건을 수정하여 동일한 리소스에 대한 중복 작업을 허용하지 않도록 했습니다.
- 스냅샷을 찾을 수 없을 때 ASA 드라이버 초기화가 실패하던 문제를 수정했습니다.
- 클러스터 노드가 기본 HA 영역에 없는 경우 지역 Flex Unified 풀에 대한 GCNV 프로비저닝이 수정되었습니다. Trident는 더 이상 해당 HA 영역을 필수 CSI 토폴로지로 처리하지 않습니다. 영역 풀은 변경되지 않았습니다. Flex Unified 구성에 대해서는 "블록 스토리지(iSCSI) 구성"를 참조하십시오.
- 정규 백엔드 ID가 Trident 볼륨 이름과 다를 경우 백엔드 조정 중에 GCNV Flex Unified NAS 및 SAN 볼륨이 잘못 고립된 것으로 표시되는 문제를 수정했습니다(이슈 "#1189"). GCNV 백엔드 구성에 대한 자세한 내용은 "Google Cloud NetApp Volumes 구성"을(를) 참조하십시오.

- 전송이 커널에 내장된 호스트에서 NVMe/TCP 감지 문제를 수정했습니다(이슈 "#1019"). 노드 요구 사항은 "작업자 노드를 준비합니다"를 참조하십시오.
- 스냅샷 예약을 포함하여 ONTAP 볼륨이 요청에 충분히 큰 상태일 때 ONTAP-NAS 및 ONTAP-NAS-FlexGroup 확장이 무한정 재시도되는 문제를 수정했습니다. 볼륨 확장에 대해서는 "볼륨 확장"를 참조하십시오.
- 접합 경로 누락으로 인해 볼륨 마운트가 실패하는 문제를 해결했습니다. ONTAP NAS 구성은 "ONTAP NAS 드라이버 구성"을(를) 참조하십시오.
- `storagePrefix`에 대시가 포함된 경우 ONTAP-SAN-Economy 볼륨 생성이 실패하던 문제를 수정했습니다. 백엔드 구성 옵션은 "ONTAP SAN 구성 옵션 및 예"을(를) 참조하십시오.
- 자동 내보내기 정책을 사용하는 읽기 전용 ONTAP-NAS 클론 게시 문제를 해결하여 원본 FlexVol 내보내기 규칙이 유지되도록 했습니다. ONTAP NAS 구성에 대한 자세한 내용은 "ONTAP NAS 구성 옵션 및 예"를 참조하십시오.
- HTTP 전송 재시도 동작이 수정되었습니다(이슈 "#1139").
- SCSI 장치 제거를 건너뛰는 고스트 장치 감지 문제를 수정했습니다(이슈 "#1161").
- 동시성 지원 프로비저닝이 선택한 노드의 기본 영역이 아닌 다른 토폴로지 영역을 선택하는 문제가 수정되었습니다. 동시성 및 토폴로지 동작에 대해서는 "컨트롤러 확장성"을(를) 참조하십시오.
- 동시성이 활성화된 경우 동일한 백엔드 간에 볼륨 배치가 불균형했던 문제가 수정되었습니다. 이제 백엔드는 비동시 프로비저닝과 마찬가지로 무작위로 균등하게 선택됩니다. 자세한 내용은 "컨트롤러 확장성"을(를) 참조하십시오.
- Trident 노드 포드 시작 중 호스트 네트워크 주소 테이블이 변경될 때 IP 주소가 누락되어 NFS 내보내기 정책 규칙이 제거될 수 있는 노드 등록 문제를 수정했습니다.
- REST 폴링 이후에도 클론 생성이 계속되어 PVC를 분할되지 않은 FlexClone으로 바인딩하는 문제를 수정했습니다. 이제 Trident는 이후의 생성 작업 실패를 감지합니다.
- ONTAP-SAN 볼륨 확장 시 노드에서 잘못된 크기 조정 실패가 보고되는 문제를 수정했습니다. 볼륨 확장에 대해서는 "볼륨 확장"을(를) 참조하십시오.

실험적 개선 사항

> 프로덕션 환경에서는 사용할 수 없습니다.

- **[기술 미리보기] Kubernetes**: ONTAP S3에서 오브젝트 스토리지 버킷의 COSI 기반 동적 프로비저닝 지원이 추가되었습니다. 자세한 내용은 "COSI를 사용하여 오브젝트 스토리지 프로비저닝"을(를) 참조하십시오.

Trident 프로텍트

향상된 기능

- AppVault 스냅샷 및 백업 메타데이터 암호화에 대한 지원이 추가되었습니다. 각 AppVault에 대해 사용자 지정 볼트 암호를 사용합니다. 볼트 암호를 교체하면 메타데이터 암호화 키가 재암호화되고 AppVault의 모든 Kopia 및 Restic 리포지토리가 업데이트됩니다. 자세한 내용은 "Trident Protect 스냅샷 및 백업용 AppVault 메타데이터 암호화"을(를) 참조하십시오.
- 복원 시 KubeVirt VM의 MAC 주소를 유지하는 기능이 추가되었습니다. 이 기능은 애플리케이션 라이선스가 MAC 주소에 연결된 경우에 유용합니다. 이 기능을 활성화하려면 protect.trident.netapp.io/retain-mac-address 어노테이션을 "true"로 설정하십시오. 자세한 내용은 "고급 Trident Protect 복원 설정 사용"을(를) 참조하십시오.
- 애플리케이션의 관련 영구 볼륨 클레임을 동일한 시점에 캡처하는 볼륨 그룹 스냅샷에 대한 지원이 추가되었습니다. 레이블, KubeVirt 가상 머신 또는 둘 다를 기준으로 애플리케이션 사용자 지정 리소스의 `spec.volumeGroups`에 볼륨 그룹을 정의합니다. 자세한 내용은 "Trident Protect에서 볼륨 그룹 스냅샷 사용"를 참조하십시오.
- 복제된 애플리케이션의 이전 스냅샷으로 장애 조치를 수행하는 기능이 추가되어 데이터가 손상되거나 유출되기

이전 시점으로 복구할 수 있습니다. 자세한 내용은 "Trident Protect를 사용하여 이전 스냅샷으로 페일오버합니다."을(를) 참조하십시오.

수정

- `trident-protect-pre-upgrade-hook` 작업에서 `trident-protect-controller-manager` ServiceAccount가 누락되어 Trident Protect 설치가 실패하는 문제를 수정했습니다("문제 #1154").
- 소유자가 존재하는 경우에만 `ownerReference`을(를) 포함하도록 PVC 전용 복원을 수정했습니다.

26.06.2의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

향상된 기능

수정

- * Kubernetes *:
  - Trident 컨트롤러가 재시작되거나 `TridentBackendConfig`이(가) 업데이트될 때 `enableConcurrency`이(가) `true`으로 설정된 상태에서 ONTAP-NAS 내보내기 정책 규칙이 자동으로 삭제되는 문제가 수정되었습니다. 이로 인해 이미 마운트된 볼륨이 고립될 수 있었습니다("문제 #1166", "문제 #1179").
  - 자격 증명 및 QoS 등 중요하지 않은 변경 사항에 대해 볼륨별 고아 스캔을 실행하여 대규모 백엔드에서 CSI 작업을 몇 시간 동안 차단할 수 있었던 백엔드 업데이트 문제를 수정했습니다. 이제 스캔은 SVM 또는 스토리지 접두사가 변경될 때만 실행됩니다.
  - CSI gRPC 요청 덤프, 시작 환경 덤프, CRD 컨트롤러에서 기록하는 `TridentBackendConfig` 사양 등 Trident 디버그 로그에 자격 증명이 나타나는 문제를 수정했습니다. 이제 시작 환경 덤프에는 Trident 자체 `TRIDENT_*` 변수 값만 기록되고 다른 변수는 이름으로 나열됩니다.

26.06.1의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

수정

- * Kubernetes *:
  - CSI 게시와 볼륨 이동 간의 경쟁 조건으로 인해 오래된 액세스 정보를 사용하여 볼륨이 연결될 수 있었던 문제를 수정했습니다.
  - NodeGetInfo가 노드 등록이 완료되기 전에 빈 토폴로지 레이블을 반환하여 토폴로지 인식 프로비저닝 실패를 유발하는 경쟁 조건을 수정했습니다.
  - NVMe 연결 해제 시 발생하는 드문 경쟁 문제를 수정했습니다. 이 문제로 인해 다른 Pod가 연결 중인 동안 공유 서브시스템이 종료될 수 있었습니다.
  - 동시 실행 환경에서 마운트한 볼륨의 클론을 생성할 때 발생하던 NVMe PVC 클론 게시 실패(및 잘못된 상위 네임스페이스 매핑) 문제를 수정했습니다.

- nameTemplate을 사용하는 ONTAP-SAN-Economy 백엔드의 PVC에 대한 스냅샷 생성 실패 문제를 수정했습니다.

향상된 기능

- * Kubernetes *:
  - ONTAP 백엔드 구성에 REST 및 ZAPI API 클라이언트 타임아웃을 백엔드별로 설정할 수 있는 선택적 `storageAPITimeout`(초)을 추가했습니다.

26.06의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

향상된 기능

- * Kubernetes *:
  - ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버에 동시성 지원을 일반(GA) 버전으로 추가하여 해당 기능을 기술 미리 보기에서 정식 기능으로 전환했으며, Azure NetApp Files(ANF) 및 SolidFire 드라이버에도 동시성 지원을 일반 버전으로 추가했습니다.
  - Kubernetes 1.36에 대한 지원이 추가되었습니다.
  - Google Cloud NetApp Volumes(GCNV) NAS 및 SAN 드라이버에서 볼륨 자동 확장에 대한 일반 공급(GA) 지원이 추가되었습니다.
  - 새 볼륨을 프로비저닝할 때 최적의 애그리게이트 선택을 위한 Unified ONTAP 균형 배치 기능이 추가되었습니다("문제 #64").
  - ONTAP-NAS-Flexgroup 드라이버에 ONTAP AFX 지원이 추가되었습니다.
  - NetApp Shift Toolkit과의 통합 기능이 추가되어 Red Hat OpenShift Migration Toolkit for Virtualization을 사용할 때 가상 머신 마이그레이션을 간소화합니다.
  - GCNV SAN 드라이버에 클라우드 ID 지원이 추가되었습니다.
  - Trident Volume Publication(TVP) 전파 및 볼륨 자동 확장 워크플로에 하위 볼륨 지원이 추가되었습니다.
  - Trident 동시 실행 모드로 실행할 때 볼륨 업데이트 및 볼륨 그룹 스냅샷 지원이 추가되었습니다.
  - GCNV에 동적 gRPC 속도 제한 기능을 추가했습니다.
  - 컨트롤러 준비 상태 실행 프로브를 추가하고 컨트롤러 및 사이드카에 대한 기본 Kubernetes API QPS 및 버스트 설정을 조정했습니다.
  - 노드 연결, 암호화 설정 및 파일 시스템 포맷 단계를 분리하여 노드 작업 순서를 개선했습니다.
  - 보안 강화를 위해 Trident 컨테이너의 루트 파일 시스템을 읽기 전용으로 마운트했습니다.
  - ONTAP 백엔드에서 클론 볼륨의 원본 스냅샷이 삭제될 때 클론 볼륨 스냅샷도 최대한 삭제되도록 하는 기능을 추가했습니다.
  - Veeam Kasten, Trilio 및 Cohesity에서 생성된 임시 PVC가 ONTAP 볼륨 복구 대기열에 갇히는 것을 방지하기 위해 백업 클론 PVC 인식 기능을 추가했습니다.
  - 특수 지역을 지원하기 위해 AWS ARN 처리 기능이 향상되었습니다.
  - ONTAP-SAN-ECONOMY 드라이버의 크기 조정 작업 중 Flexvol이 축소되는 것을 방지합니다.

실험적 개선 사항

프로덕션 환경에서는 사용할 수 없습니다.

- **[Tech Preview]**: ONTAP-NAS-Economy qtree 가져오기 지원이 추가되었습니다.
- **[Tech Preview]**: ASA-R2(SAN 및 NVMe) 드라이버에 동시성 지원이 추가되었습니다.
- **[Tech Preview]**: ONTAP-SAN 및 ONTAP-NAS 드라이버에 GCNV ONTAP 모드 HTTP 전송 기능이 추가되었습니다.

수정

- * Kubernetes *:
  - 게시/게시 취소, 클론, 캐시 및 백엔드 업데이트 작업의 동시 실행에 대한 경쟁 조건 및 상태 처리 문제를 수정했습니다.
  - ONTAP-NAS 및 ONTAP-NAS-Economy 드라이버에서 하위 볼륨 및 읽기 전용 클론에 대한 내보내기 정책 경쟁 조건과 동시 게시/게시 취소 처리 문제를 수정했습니다.
  - 스냅샷 정책이 다를 때("문제 #1082") MetroCluster 페일오버 후 ONTAP-NAS-Economy FlexVol 검색 문제를 수정했습니다.
  - 장시간 실행되는 qtree 삭제를 중단시키는 ONTAP-NAS-Economy 볼륨 삭제 재시도 문제를 수정했습니다("문제 #1121").
  - ONTAP-SAN 및 ONTAP-SAN-Economy의 가져오기 및 크기 조정 동작이 수정되었습니다. 여기에는 fsType 유효성 검사, 볼륨 메타데이터 처리 및 자동 증가 모드 동작이 포함됩니다.
  - 동시 볼륨 생성 중 NVMe/TCP 네임스페이스 경쟁 조건을 수정했습니다("문제 #1089").
  - 사용 가능한 경로와 포털이 일치하지 않을 때 발생하던 iSCSI 크기 조정 실패 문제를 수정하고, 반복적인 크기 조정 동작을 개선했습니다.
  - 볼륨 사용 중 상태로 인해 발생하던 ONTAP-SAN 볼륨 삭제 오류를 수정했습니다.
  - ONTAP REST 오류 처리 문제를 수정하고 SVM 상태 타임아웃 동작을 개선했습니다.
  - GCNV SAN 영역 풀 및 호스트 그룹 처리 문제를 수정하고, 볼륨 식별자 변환을 표준화했으며, 최소 크기 미만의 생성 요청을 표준화했습니다.
  - GCNV 네이티브 용량 풀 선택 시 ONTAP 모드 풀을 제외하도록 수정했습니다.
  - KubeVirt Trident 자동 워크로드 페일오버를 사용하는 VM에 대한 KubeVirt 페일오버 문제를 수정하여 VM 페일오버 속도를 향상시켰습니다.
  - Operator 관리 설치 RBAC 문제(누락된 노드 ClusterRoles 및 업그레이드 중 잘못된 ClusterRole 및 ClusterRoleBinding 조정 오류 포함)를 수정했습니다. 이 수정 사항은 Trident Operator 차트를 사용하는 Operator 및 Helm 기반 설치에 적용됩니다.
  - 기본 우선순위 값을 생략하고 YAML 템플릿 들여쓰기를 수정하여 TridentConfigurator 및 관련 CRD에서 지속적으로 발생하던 GitOps diff 문제를 해결했습니다("문제 #915", "문제 #1120").
  - Helm 업그레이드 중 TridentVolumePublication CRD의 패치 수정("문제 #1151").
  - 노드 등록 재시도 전에 CSI gRPC 서비스를 시작하여 CSI 시작 순서를 수정했습니다.
  - 동시 작업 중 보안 SMB 클론 구성 오버레이 문제를 수정했습니다.
  - Trident 업그레이드 후 Trident 컨트롤러 포드에서 누락된 주석을 수정했습니다("문제 #1004").

- 대형 SVM에서 느린 qtree 프로비저닝이 개선되었습니다("문제 #1142").
- ControllerPublish가 잘못된 기본 파일 시스템 유형 대신 볼륨 구성 파일 시스템 유형을 사용하도록 수정했습니다.
- 고아 dm 장치로 인해 CSI NodeUnstageVolume 오류가 반복적으로 발생하는 문제를 수정했습니다.
- 소프트 삭제된 노드 정리 후 고스트 다중 경로 아티팩트와 오래된 TridentNode 리소스를 제거하여 오래된 노드 및 다중 경로 정리 기능을 개선했습니다.

Trident 프로텍트

향상된 기능

- OpenShift Service CA Operator 통합 기능을 추가하여 Trident Protect 승인 웹훅에 대한 TLS 인증서 관리를 자동화합니다. Helm 값에서 `useOpenshiftServiceCA`을(를) `true`로 설정하여 활성화합니다. 자세한 내용은 "Trident Protect 설치 사용자 정의"을 참조하십시오.
- KubeVirt VM을 OpenShift 가상화 환경에서 백업 및 복원하는 기능이 추가되었습니다. 네임스페이스 내의 모든 VM을 보호하거나 이름으로 특정 VM을 대상으로 지정하여 백업 또는 스냅샷에서 개별 VM을 선택적으로 복원할 수 있습니다. 자세한 내용은 "KubeVirt 가상 머신을 Trident Protect로 보호합니다"을 참조하십시오.
- 리소스 변환 지원이 추가되어 복원 프로세스 중에 Kubernetes 리소스 속성을 추가, 제거 및 수정할 수 있습니다. 자세한 내용은 "사용자 지정 리소스를 사용하여 리소스를 복원할 때 리소스를 수정합니다"을 참조하십시오.
- `storageClassMapping`에 대한 지원이 AppMirrorRelationship(AMR)에 추가되어 단일 AMR 정의 내에서 개별 PVC에 대한 스토리지 클래스 매핑이 가능해졌습니다. 자세한 내용은 "Trident Protect를 사용하여 애플리케이션 복제를 설정하고 관리합니다."을 참조하십시오.

수정

- 소스 애플리케이션 네임스페이스가 타겟 클러스터에 존재하지 않을 때 승인 웹훅 오류로 인해 클러스터 간 가상 머신 복원이 실패하는 문제를 수정했습니다.
- 기존 애플리케이션의 사용자 지정 리소스와의 사양 불일치로 인해 애플리케이션 생성 단계에서 실패하던 선택적 가상 머신 제자리 복원 문제를 수정했습니다. 이제 기존 애플리케이션이 재사용됩니다.
- 가상 머신과 해당 PersistentVolumeClaim이 모두 삭제된 경우 가상 머신 제자리 복원 작업이 `Running` 상태에서 중단되는 문제를 수정했습니다. 누락된 PersistentVolumeClaim은 이제 치명적인 오류로 처리됩니다.
- 기존 보호 일정을 편집할 때 백업 또는 스냅샷이 즉시 실행되는 오류가 수정되었습니다.
- 스냅샷 최종화 도구가 애플리케이션 사용자 지정 리소스가 삭제된 후에도 해당 리소스를 가져오려고 시도하는 문제를 수정했습니다. 이제 최종화 도구는 애플리케이션을 찾을 수 없는 경우 해당 리소스를 건너뜁니다.
- 스파스 복사를 사용하는 PersistentVolumeClaim 복사 단계에서 가상 머신 인플레이스 복원 후 고정 보호 스토리지 용량이 크게 증가하는 문제가 수정되었습니다.
- ONTAP의 백업 스냅샷 삭제를 확인하기 전에 Kubernetes 스냅샷 객체를 삭제하던 Trident Protect 문제를 수정했습니다. 이제 파이널라이저는 `VolumeSnapshot` 삭제를 기다립니다.
- 가상 머신 동결 작업 실패 시 스냅샷이 진행되지 않던 문제를 수정했습니다. 이제 동결 및 동결 해제 작업이 실패하더라도 계속 진행됩니다.
- AppMirrorRelationship 역방향, 역방향 재동기화 및 재동기화 작업 중에 KubeVirt 가상 머신 애플리케이션이 PersistentVolumeClaims를 잘못 재생성하는 문제를 해결했습니다.

26.02.1의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

수정

- * Kubernetes *:
- 블록(iSCSI) 워크로드에 대해 Flex Unified ZONAL 풀을 지원하도록 Google Cloud NetApp Volumes(GCNV) SAN 동작이 수정되었습니다. 이제 영역 및 지역 구성이 모두 지원됩니다.
- Google Cloud NetApp Volumes(GCNV) NAS UNIFIED 풀 처리 문제를 개선하여 볼륨 생성, 유효성 검사, 이름 지정 및 `InternalID`이 없을 때 대체 볼륨 조회 작업을 향상시켰습니다.
- 사용 가능한 경로와 포털이 일치하지 않을 때 iSCSI 디바이스 크기 조정 작업이 실패할 수 있었던 문제를 수정했습니다.
- `blkid`가 LUKS 장치의 파일 시스템 유형을 잘못 식별하여 새 볼륨 생성을 방해하던 간헐적인 문제를 수정했습니다.
- 대규모 LUKS 게시 및 스테이징 작업으로 인해 Trident 컨트롤러가 차단되는 문제를 수정했습니다. 이제 사용자는 볼륨과 연결된 LUKS 암호를 추적해야 합니다.
- iSCSI 다중 경로 파티션이 고스트 장치로 나타나 CSI `NodeUnstage` 작업이 성공하지 못하는 문제를 수정했습니다.
- 동시 볼륨 생성 중 NVMe/TCP 네임스페이스 경합 조건을 수정했습니다(Issue #1089).
- 스냅샷이 같은 초 내에 실행될 때 발생했던 동시 클론 스냅샷 작업 문제를 해결했습니다.
- `dataVolumes`를 사용하는 KubeVirt VM이 Trident 자동 워크로드 페일오버로 페일오버되지 않던 문제를 수정했습니다.
- Trident 동시 실행 코어를 사용하는 동안 유효하지 않은 `TridentMirrorRelationship` 사용자 지정 리소스를 삭제할 때 발생하던 오류를 수정했습니다.
- `tridentactionmirrorupdates` Kubernetes CRD YAML 정의의 들여쓰기 문제를 수정했습니다(이슈 #1120).
- REST qtree 비동기 삭제 API 호출 문제를 수정했습니다(Issue #1121).
- Fixed `ControllerPublish` 잘못된 기본 파일 시스템 유형 대신 볼륨 구성 파일 시스템 유형을 사용하도록 수정했습니다.
- 컨트롤러 시작 로깅이 동시 드라이버 상태를 올바르게 보고하도록 수정했습니다.
- 동시 실행 Trident에서 일시 중단된 백엔드 처리를 수정했습니다.
- 특수 지역을 지원하기 위해 AWS ARN 처리 기능이 향상되었습니다.

Trident 프로텍트

수정

보안 취약점을 해결하기 위해 Trident Protect 종속성이 업데이트되었습니다.

- gRPC-Go: "CVE-2026-33186"
- Go x/crypto SSH: "CVE-2024-45337"

- OpenTelemetry-Go: "CVE-2026-24051"
- Go 런타임: "CVE-2025-22871"
- 추가 종속성: "CVE-2025-23109"

26.02의 새로운 기능

NetApp Trident 및 Trident Protect의 새로운 기능, 개선 사항, 버그 수정 및 사용 중단 예정 기능에 대해 알아보세요.

트라이던트

향상된 기능

- * Kubernetes *:
  - ONTAP-NAS(NFS만 해당), ONTAP-SAN(iSCSI, FCP, NVMe) 및 Google Cloud NetApp Volumes(GCNV) 드라이버에 대한 동시성 지원이 정식 출시(GA)되어 이 기능이 Tech Preview에서 졸업했습니다. 자세한 내용은 "컨트롤러 확장성 문서"을 참조하십시오.
  - 사용자 정의 Trident AutoGrow 정책을 기반으로 Trident의 볼륨 자동 확장 지원이 추가되었습니다. 자세한 내용은 "자동 볼륨 확장 설명서"를 참조하십시오.
  - NVMe 볼륨에 대한 노드 작업의 확장성을 향상시키기 위해 Trident 노드 동시성이 강화되었습니다. 자세한 내용은 "NVMe 볼륨 문서"를 참조하십시오.
  - PVC 주석을 통한 GCNV NAS 볼륨 자동 계층화 지원 추가(`tieringPolicy`, `tieringMinimumCoolingDays`, 풀 선택 및 클론 상속 포함. 자세한 내용은 "Google Cloud NetApp Volumes에 대한 자동 계층화 구성 설명서"을(를) 참조하십시오.
  - `google-cloud-netapp-volumes-san` 드라이버를 사용하여 GCNV SAN 블록(iSCSI) 볼륨에 대한 지원이 추가되었습니다. 여기에는 프로비저닝, LUN 액세스를 위한 노드별 호스트 그룹 매핑 및 볼륨 복제 작업이 포함됩니다. 자세한 내용은 "Google Cloud NetApp Volumes 블록 구성 설명서"를 참조하십시오.
  - Amazon FSx for NetApp ONTAP에 대한 자동 백엔드 구성 지원이 추가되었습니다. 필수 매개변수를 포함하는 StorageClass를 생성하면 Trident는 필요한 경우 해당 백엔드와 VolumeSnapshotClass를 자동으로 생성합니다. 자세한 내용은 "Amazon FSx for NetApp ONTAP 백엔드 구성 설명서"를 참조하십시오.
  - Azure Government 및 Azure China와 같은 다양한 Microsoft Azure 클라우드에 대한 지원이 추가되었으며, Azure NetApp Files 백엔드에 대한 사용자 지정 클라우드 구성도 추가되었습니다. 자세한 내용은 "Azure NetApp Files 백엔드 구성 설명서"을 참조하십시오.
  - Kubernetes 1.35에 대한 지원이 추가되었습니다. 자세한 내용은 "요구사항 문서"을 참조하십시오.

실험적 개선 사항

프로덕션 환경에서는 사용할 수 없습니다.

- **[Tech Preview]**: ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버에 대한 동시성 지원이 추가되었습니다.

알려진 문제

`csi-snapshotter` 사이드카에 알려진 문제가 있습니다. 모든 Kubernetes 버전에서 `VolumeGroupSnapshots` `v1beta1`는 `VolumeSnapshots`가

`ReadyToUse` 상태에 도달하는 것을 방지합니다.

두 가지 해결 방법이 있습니다.

1. VolumeGroupSnapshots를 비활성화하려면 VolumeGroupSnapshots CRD를 삭제한 다음 Trident를 다시 설치합니다.
2. VolumeGroupSnapshots `v1beta2` 및 `snapshot-controller` 버전 8.4.0 이상을 설치한 다음 Trident를 다시 설치합니다. VolumeGroupSnapshots는 Kubernetes 버전 v1.34 이전 버전에서는 작동하지 않습니다.

수정

- * Kubernetes *:
  - ONTAP-NAS, ONTAP-NAS-Economy 및 Google Cloud NetApp Volumes 드라이버에서 읽기 전용 클론을 게시 취소하면 소스 볼륨에서 내보내기 정책 규칙이 제거되는 문제를 수정했습니다"문제 #1086".
  - Bitnami의 공개 이미지 사용 중단에 따른 풀 실패를 방지하기 위해 kubectl 이미지를 경량 Alpine 기반 변형으로 전환했습니다 "문제 #1080".
  - Trident 업그레이드 중 기존 배포의 주석 보존 문제를 수정했습니다"문제 #1004".
  - 두 스토리지 클래스가 동일한 백엔드를 가리키는 경우 서로 다른 스토리지 클래스 간의 클론 생성을 허용합니다 "문제 #1104".
  - 클라우드 환경에서 네트워크 지연으로 인한 시간 초과로 발생한 노드 준비 실패를 수정했습니다. 클라우드 기반 설치에 대한 시간 초과 값을 늘렸습니다.
  - LUN 생성 시 프로세스가 재시도 상태에 진입했을 때 파일 시스템 유형 속성이 설정되지 않은 상태로 남아 있던 문제를 수정했습니다.
  - REST API 볼륨 조회 시 볼륨 상태를 무시하도록 수정하여 볼륨 쿼리 중 발생하는 오탐을 방지했습니다.
  - 대규모 환경에서 ontap-nas-economy 드라이버를 사용할 때 Trident 컨트롤러의 효율성이 향상되었습니다.
  - ontap-san-economy 드라이버에서 LUN 가져오기 중에 internalID를 설정합니다.
  - 더 많은 서브넷을 처리할 수 있도록 Azure Resource Graph 쿼리 제한이 증가했습니다.
  - 일부 백업 애플리케이션에서 경쟁 조건을 방지하기 위해 CSI 및 ONTAP 클론 분할 시간 제한이 개선되었습니다"문제 #1098", "문제 #1100".
  - LUKS 오류 메시지 억제 문제를 해결했습니다"문제 #1069".
  - iSCSI 및 NVMe 프로토콜 모두에서 오래된 LUKS 매퍼 처리 문제를 수정했습니다. 향상된 정리 로직을 통해 사용되지 않는 장치 매퍼로 인한 마운트 오류를 방지합니다.
- RWX NVMe 볼륨에 대한 확장 제한 사항을 수정했습니다.
- OpenTelemetry-Go 패키지를 업데이트하여 "CVE-2026-24051"를 해결했습니다.

Trident 프로텍트

향상된 기능

- Trident Protect는 이제 제자리 복원 전에 보호 일정을 자동으로 비활성화하고 진행 중인 작업을 취소한 다음 복원이 완료된 후 다시 활성화합니다. 자세한 내용은 "Trident Protect를 사용하여 애플리케이션 복원"를 참조하십시오.
- 스케줄 생성 시 즉시 백업 또는 스냅샷을 트리거하기 위해 스케줄 CR에 `runImmediately` 필드를 추가하고

`--run-immediately` CLI 플래그를 추가했습니다. 자세한 내용은 "데이터 보호 스케줄을 생성합니다"을(를) 참조하십시오.

- 복원 CR의 `destinationApplicationName` 필드 또는 `--destination-app-name` CLI 플래그를 사용하여 복원된 애플리케이션에 대한 사용자 지정 이름 지정 지원이 추가되었습니다. 자세한 내용은 "Trident Protect를 사용하여 애플리케이션 복원"을(를) 참조하십시오.

수정

- 필요한 서비스 계정을 사용할 수 있게 되기 전에 Pod가 생성되어 발생하던 복원 실패 문제를 수정했습니다.
- `Roles` 및 `RoleBindings`가 애플리케이션 복원 중에 건너뛰어지던 문제가 수정되었습니다.
- 올바르게 구성되었음에도 불구하고 `tridentctl-protect get appvaultcontent` 출력에 원래 클러스터 이름이 표시되지 않던 문제를 수정했습니다.
- `pipefail` 처리 누락으로 인해 무시되었던 Kopia 복원 오류를 수정했습니다.
- 영구 볼륨을 제외하는 리소스 필터로 인해 발생했던 스냅샷 및 백업 오류를 수정했습니다.
- 네임스페이스 간에 동일한 이름의 PVC가 있는 다중 네임스페이스 애플리케이션에서 잘못된 PVC 복원으로 인해 데이터 손실이 발생할 수 있었던 문제를 수정했습니다.

25.10의 새로운 기능

Trident 와 Trident Protect의 새로운 기능, 개선 사항, 수정 사항, 지원 중단 사항을 알아보세요.

트라이던트

향상된 기능

- * Kubernetes *:
  - ONTAP -SAN(iSCSI 및 FC) 외에도 ONTAP-NAS NFS 및 ONTAP -SAN-Economy 드라이버에 대한 v1beta1 볼륨 그룹 스냅샷 Kubernetes API를 통해 CSI 볼륨 그룹 스냅샷에 대한 지원이 추가되었습니다. 보다"볼륨 그룹 스냅샷 작업" .
  - ONTAP-NAS 및 ONTAP-NAS-Economy(두 NAS 드라이버 모두에서 SMB 제외)와 ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버에 대해 강제 볼륨 분리를 통한 자동 워크로드 장애 조치 지원이 추가되었습니다. 보다"Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화" .
  - FCP 볼륨의 노드 작업에서 더 높은 확장성을 위해 향상된 Trident 노드 동시성을 제공합니다.
  - ONTAP NAS 드라이버에 대한 ONTAP AFX 지원이 추가되었습니다. 보다"ONTAP NAS 구성 옵션 및 예" .
  - TridentOrchestrator CR 및 Helm 차트 값을 통해 Trident 컨테이너에 대한 CPU 및 메모리 리소스 요청과 제한을 구성하는 지원이 추가되었습니다. ("1000호" ,"927호" ,"853호" ,"592호" ,"110호").
  - ASAr2 개성에 대한 FC 지원이 추가되었습니다. 보다"ONTAP SAN 구성 옵션 및 예" .
  - HTTP 대신 HTTPS로 Prometheus 메트릭을 제공하는 옵션이 추가되었습니다. 보다"Trident 모니터링" .
  - 옵션을 추가했습니다 `--no-rename` 볼륨을 가져올 때 원래 이름은 유지하면서 Trident 볼륨의 수명 주기를 관리하도록 합니다. 보다"볼륨 가져오기" .
  - Trident 배포는 이제 시스템 클러스터 중요 우선 순위 클래스에서 실행됩니다.
- Trident 컨트롤러가 helm, operator 및 tridentctl을 통해 호스트 네트워킹을 사용할 수 있는 옵션을 추가했습니다 ."858호").

- 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

> 프로덕션 환경에서는 사용할 수 없습니다.

- **[**기술 미리보기**]**: ONTAP-SAN 드라이버(통합 ONTAP 9의 iSCSI 및 FCP 프로토콜)에 대한 기존 기술 미리보기 외에도 ONTAP-NAS(NFS 전용) 및 ONTAP-SAN(통합 ONTAP 9의 NVMe)에 대한 동시성 지원이 추가되었습니다.

수정

- 쿠버네티스:
  - Linux DaemonSet을 node-driver-registrar로 표준화하여 Windows DaemonSet과 컨테이너 이미지 명명 방식을 일치시켜 CSI node-driver-registrar 컨테이너 이름 불일치 문제를 해결했습니다.
  - 레거시 Qtree에 대한 내보내기 정책이 제대로 업그레이드되지 않는 문제를 해결했습니다.
- 오픈시프트:
  - SCC에서 allowHostDirVolumePlugin이 false로 설정되어 Openshift의 Windows 노드에서 고정 Trident 노드 포드가 시작되지 않는 문제가 해결되었습니다."950호").
- Helm을 통해 설정되지 않는 Kubernetes API QPS가 고정됨("975호").
- 동일한 Kubernetes 노드에서 NVMe 기반 XFS 파일 시스템 PVC의 스냅샷을 기반으로 영구 볼륨 클레임(PVC)을 마운트할 수 없는 문제를 수정했습니다.
- 백엔드별로 고유/공유 서브시스템 이름(예: netappdvp_subsystem)을 추가하여 NDVP 모드에서 호스트/Docker를 재시작한 후 발생하는 UUID 변경 문제를 해결했습니다.
- Trident 23.10 이전 버전에서 24.10 이상 버전으로 업그레이드하는 동안 발생하는 iSCSI 볼륨의 마운트 오류를 수정하여 "잘못된 SANType" 문제를 해결했습니다.
- Trident 컨트롤러를 다시 시작하지 않고도 Trident 백엔드 상태가 온라인/오프라인으로 전환되지 않는 문제가 해결되었습니다.
- PVC 크기 조절 속도가 느려지는 간헐적 경쟁 조건이 수정되었습니다.
- 볼륨 복제 실패 시 고정 스냅샷이 정리되지 않는 문제를 해결했습니다.
- 커널에 의해 장치 경로가 변경되었을 때 볼륨의 스테이지 해제에 실패하는 문제가 수정되었습니다.
- LUKS 장치가 이미 닫혀서 볼륨을 스테이지 해제하지 못하는 문제가 수정되었습니다.
- 느린 저장 작업으로 인해 ContextDeadline 오류가 발생하는 문제가 해결되었습니다.
- Trident Operator는 구성 가능한 k8s-timeout을 기다려 Trident 버전을 확인합니다.

Trident 프로텍트

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

향상된 기능

- 일정 및 백업 CR에 대한 스냅샷 CR 시간 초과를 제어하기 위한 주석이 추가되었습니다.
  - `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`

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

25.06.2의 변경 사항

트라이던트

수정

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

25.06.1의 변경 사항

트라이던트

> 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

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

25.06의 변경 사항

트라이던트

향상된 기능

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

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

  - iSCSI 외에도 NVMe/TCP용 ONTAP ASA r2에 대한 지원이 추가되었습니다. 보다"ONTAP SAN 구성 옵션 및 예" .
  - 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. 을 "보안 SMB 활성화 "참조하십시오.
  - iSCSI 볼륨의 노드 작업에서 더 높은 확장성을 제공하기 위해 향상된 Trident 노드 동시성을 제공합니다.
  - 추가됨 `--allow-discards` LUKS 볼륨을 열어 공간 회수를 위한 삭제/TRIM 명령을 허용할 때.
  - LUKS로 암호화된 볼륨을 포맷할 때 성능이 향상되었습니다.
  - 오류가 발생했지만 부분적으로 포맷된 LUKS 장치에 대한 향상된 LUKS 정리 기능입니다.
  - NVMe 볼륨 연결 및 분리를 위한 향상된 Trident 노드 멱등성.
  - 추가됨 `internalID` ONTAP-SAN-Economy 드라이버에 대한 Trident 볼륨 구성에 대한 필드입니다.
  - NVMe 백엔드에 SnapMirror를 사용한 볼륨 복제 지원이 추가되었습니다. 을 "SnapMirror를 사용하여 볼륨을 복제합니다"참조하십시오.

실험적 개선 사항

프로덕션 환경에서는 사용할 수 없습니다.

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

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

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

수정

- * Kubernetes *:
  - 기본 SCSI 디스크를 사용할 수 없는 경우 멀티패스 장치의 크기가 일치하지 않는 CSI NodeExpandVolume의 문제를 해결했습니다.
  - ONTAP-NAS 및 ONTAP-NAS-Economy 드라이버에 대한 중복된 내보내기 정책을 정리하지 못하는 문제가 수정되었습니다.
  - `nfsMountOptions`가 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 고정된 배포 문제("문제 #831").
- 스냅샷에서 생성된 PVC에 대한 누락된 내보내기 정책이 수정되었습니다. "문제 #1016").
- ANF 볼륨 크기가 1GiB 단위로 자동 정렬되지 않는 문제가 해결되었습니다.
- Bottlerocket과 함께 NFSv3를 사용할 때 발생하는 문제가 해결되었습니다.

- 크기 조정에 실패했음에도 불구하고 ONTAP-NAS-Economy 볼륨이 최대 300TB까지 확장되는 문제가 해결되었습니다.
- ONTAP REST API를 사용할 때 복제 분할 작업이 동기적으로 수행되는 문제가 해결되었습니다.

사용 중단:

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

Trident 프로텍트

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

향상된 기능

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

수정

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

25.02.1의 변경 사항

트라이던트

수정

- * Kubernetes *:
  - 기본 이미지 레지스트리()를 사용할 때 사이드카 이미지 이름 및 버전이 잘못 채워지는 Trident-연산자의 문제를 해결했습니다"문제 #983".
  - ONTAP 페일오버 반환 중에 다중 경로 세션이 복구되지 않는 문제를 해결했습니다"문제 #961"( ).

25.02의 변경 사항

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

트라이던트

향상된 기능

- * Kubernetes *:
  - iSCSI에 대한 ONTAP ASA R2 지원이 추가되었습니다.
  - 비원활한 노드 종료 시나리오 중 ONTAP-NAS 볼륨에 대한 강제 분리 지원 추가 새로운 ONTAP-NAS 볼륨은 이제 Trident에서 관리하는 볼륨별 엑스포트 정책을 활용합니다. 활성 워크로드에 영향을 주지 않고 게시 취소 시 기존 볼륨을 새로운 엑스포트 정책 모델로 전환할 수 있는 업그레이드 경로를 제공했습니다.
  - cloneFromSnapshot 주석이 추가되었습니다.
  - 교차 네임스페이스 볼륨 클론 복제에 대한 지원이 추가되었습니다.
  - 정확한 호스트, 채널, 타겟 및 LUN ID를 기준으로 재검색을 시작하기 위한 향상된 iSCSI 자동 복구 검사 수정 기능
  - Kubernetes 1.32에 대한 지원 추가
- * OpenShift *:
  - ROSA 클러스터에서 RHCOS에 대한 자동 iSCSI 노드 준비 지원 추가
  - OpenShift Virtualization for ONTAP 드라이버에 대한 지원이 추가되었습니다.
- ONTAP-SAN 드라이버에 대한 파이버 채널 지원이 추가되었습니다.
- NVMe LUKS 지원이 추가되었습니다.
- 모든 기본 이미지의 스크래치 이미지로 전환되었습니다.
- iSCSI 세션이 로그인되어야 하지만 () 로그인되지 않은 경우 iSCSI 연결 상태 검색 및 로깅이 추가되었습니다"문제 #961".
- Google-cloud-NetApp-volumes driver를 사용하여 SMB 볼륨에 대한 지원을 추가했습니다.
- 삭제 시 ONTAP 볼륨이 복구 대기열을 건너뛸 수 있도록 지원 추가
- 태그 대신 SHAS를 사용하여 기본 이미지를 재정의할 수 있는 지원이 추가되었습니다.
- tridentctl 설치 프로그램에 image-pull-secrets 플래그가 추가되었습니다.

수정

- * Kubernetes *:
  - 자동 엑스포트 정책에서 누락된 노드 IP 주소를 수정했습니다"문제 #965"().
  - ONTAP-NAS-Economy의 경우 조기에 볼륨별 정책으로 전환되는 고정 자동 엑스포트 정책
  - 사용 가능한 모든 AWS ARN 파티션을 지원하기 위해 백엔드 구성 자격 증명을 수정했습니다("문제 #913").
  - Trident 연산자에서 자동 구성기 조정을 비활성화하는 옵션이 추가되었습니다("문제 #924").
  - CSI-resizer 컨테이너에 대한 securityContext가 추가되었습니다("문제 #976").

Trident 프로텍트

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

향상된 기능

- volumeMode: File 및 volumeMode: Block(원시 장치) 스토리지 모두에 대해 KubeVirt/OpenShift 가상화 VM에 대한 백업 및 복원 지원이 추가되었습니다. 이 지원은 모든 Trident 드라이버와 호환되며, Trident Protect와 함께 NetApp SnapMirror 사용하여 스토리지를 복제할 때 기존 보호 기능을 향상시킵니다.
- Kubevirt 환경의 애플리케이션 수준에서 동결 동작을 제어하는 기능이 추가되었습니다.
- AutoSupport 프록시 연결 구성에 대한 지원이 추가되었습니다.
- Data Mover 암호화에 대한 암호를 정의하는 기능이 추가되었습니다(Kopia/Restic).
- 실행 후크를 수동으로 실행할 수 있는 기능이 추가되었습니다.
- Trident Protect 설치 중에 보안 컨텍스트 제약 조건(SCC)을 구성하는 기능이 추가되었습니다.
- Trident Protect 설치 중 nodeSelector 구성에 대한 지원이 추가되었습니다.
- AppVault 개체에 대한 HTTP/HTTPS 송신 프록시에 대한 지원이 추가되었습니다.
- 확장 ResourceFilter - 클러스터 범위 리소스를 제외할 수 있도록 설정합니다.
- S3 AppVault 자격 증명에서 AWS 세션 토큰에 대한 지원이 추가되었습니다.
- 사전 스냅샷 실행 후크 후 리소스 수집 지원 추가

수정

- 임시 볼륨 관리를 개선하여 ONTAP 볼륨 복구 대기열을 건너뛰었습니다.
- 이제 SCC 주석이 원래 값으로 복원됩니다.
- 병렬 작업 지원으로 복원 효율성 향상
- 대규모 애플리케이션의 실행 후크 시간 초과에 대한 지원이 향상되었습니다.

24.10.1의 변경 사항

향상된 기능

- * Kubernetes *: Kubernetes 1.32에 대한 지원이 추가되었습니다.

- iSCSI 세션이 로그인되어야 하지만 () 로그인되지 않은 경우 iSCSI 연결 상태 검색 및 로깅이 추가되었습니다"문제 #961".

수정

- 자동 엑스포트 정책에서 누락된 노드 IP 주소를 수정했습니다"문제 #965"().
- ONTAP-NAS-Economy의 경우 조기에 볼륨별 정책으로 전환되는 고정 자동 엑스포트 정책
- CVE-2024-45337 및 CVE-2024-45310을 해결하기 위해 Trident 및 Trident-ASUP 종속성이 업데이트되었습니다.
- iSCSI 자동 복구 중 간헐적으로 비CHAP 포털에 대한 로그아웃을 제거했습니다"문제 #961"().

24.10의 변경 사항

향상된 기능

- Google Cloud NetApp Volumes 드라이버는 현재 NFS 볼륨에 일반적으로 사용할 수 있으며 영역 인식 프로비저닝을 지원합니다.
- GCP 워크로드 ID는 GKE에서 Google Cloud NetApp 볼륨의 Cloud Identity로 사용됩니다.
- `formatOptions`사용자가 LUN 형식 옵션을 지정할 수 있도록 ONTAP-SAN 및 ONTAP-SAN-Economy 드라이버에 구성 매개 변수를 추가했습니다.
- Azure NetApp Files의 최소 볼륨 크기가 50GiB로 감소합니다. Azure의 새로운 최소 크기는 11월에 일반적으로 제공될 예정입니다.
- ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버를 기존 FlexVol 풀로 제한하기 위한 구성 매개 변수가 추가되었습니다 `denyNewVolumePools`.
- 모든 ONTAP 드라이버에서 SVM에서 애그리게이트의 추가, 제거 또는 이름 변경에 대한 감지가 추가되었습니다.
- 보고된 PVC 크기를 사용할 수 있도록 LUKS LUN에 18MiB 오버헤드를 추가했습니다.
- ONTAP-SAN 및 ONTAP-SAN-Economy 노드 단계 및 무단계 오류 처리를 개선하여 실패 단계 후 장치를 제거할 수 있도록 했습니다.
- 고객이 ONTAP에서 Trident의 최소한의 역할만 수행할 수 있도록 사용자 지정 역할 생성기가 추가되었습니다.
- 문제 해결을 위한 추가 로깅이 `lsscsi` 추가되었습니다("문제 #792").

쿠버네티스

- Kubernetes 네이티브 워크플로우를 위한 새로운 Trident 기능 추가:
  - 데이터 보호
  - 데이터 마이그레이션
  - 재해 복구
  - 애플리케이션 이동성

    "Trident Protect에 대해 자세히 알아보세요".
- 새로운 플래그를 추가했습니다 `--k8s-api-qps` Trident 가 Kubernetes API 서버와 통신하는 데 사용하는 QPS 값을 설정하도록 설치 프로그램에 요청합니다.

- Kubernetes 클러스터 노드에서 스토리지 프로토콜 종속성을 자동으로 관리하기 위한 설치 관리자에 플래그가 추가되었습니다 `--node-prep`. Amazon Linux 2023 iSCSI 스토리지 프로토콜과의 호환성 테스트를 거쳐 검증되었습니다
- 비원활한 노드 종료 시나리오 중 ONTAP-NAS-Economy 볼륨에 대한 강제 분리 지원 추가
- 새로운 ONTAP-NAS-Economy NFS 볼륨은 백엔드 옵션을 사용할 때 qtree별 엑스포트 정책을 `autoExportPolicy` 사용합니다. qtree는 게시 시점에 노드 제한 엑스포트 정책에만 매핑되어 액세스 제어와 보안을 개선합니다. Trident이 활성 워크로드에 영향을 주지 않고 모든 노드에서 볼륨을 게시 해제하면 기존 qtree가 새로운 엑스포트 정책 모델로 전환됩니다.
- Kubernetes 1.31에 대한 지원 추가

실험적 개선 사항

- ONTAP-SAN 드라이버에서 파이버 채널 지원에 대한 기술 미리 보기가 추가되었습니다.

수정

- * Kubernetes *:
  - Trident Helm 설치를 방지하는 고정 레인처 입원 웹후크"문제 #839"().
  - helm chart 값의 Affinity 키를 수정했습니다"문제 #898"().
  - 고정 tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector는 "true" 값()과 함께 작동하지 "문제 #899" 않습니다.
  - 클론 생성 중에 생성된 임시 스냅샷을 삭제했습니다"문제 #901"( ).
- Windows Server 2019에 대한 지원이 추가되었습니다.
- Trident repo()의 'Go mod queret' 수정"문제 #767"

사용 중단

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

제품 브랜드 변경

24.10 릴리즈부터 Astra Trident은 NetApp Trident(Trident)으로 브랜드를 변경했습니다. 이 브랜드 변경은 Trident의 기능, 지원되는 플랫폼 또는 상호 운용성에 영향을 미치지 않습니다.

24.06의 변경 사항

향상된 기능

- 중요: `limitVolumeSize` 이제 매개 변수는 ONTAP 이코노미 드라이버에서 qtree/LUN 크기를 제한합니다. 새 매개변수를 사용하여 `limitVolumePoolSize` 해당 드라이버에서 FlexVol 크기를 제어합니다. "문제 #341"().
- 더 이상 사용되지 않는 igroup이 사용 중인 경우 정확한 LUN ID로 SCSI 검색을 시작하는 iSCSI 자동 복구 기능이 추가되었습니다("문제 #883").
- 백엔드가 일시 중단 모드인 경우에도 볼륨 클론 및 크기 조정 작업에 대한 지원이 추가되었습니다.

- Trident 컨트롤러에 대해 사용자 구성 로그 설정이 Trident 노드 포드로 전파되는 기능이 추가되었습니다.
- ONTAP 버전 9.15.1 이상에서 ONTAPI(ZAPI) 대신 기본적으로 REST를 사용하도록 Trident에 지원이 추가되었습니다.
- ONTAP 스토리지 백엔드에서 새로운 영구 볼륨에 대해 맞춤형 볼륨 이름 및 메타데이터에 대한 지원 추가
- `azure-netapp-files`NFS 마운트 옵션이 NFS 버전 4.x를 사용하도록 설정된 경우 기본적으로 스냅샷 디렉토리를 자동으로 사용하도록 (ANF) 드라이버를 개선했습니다
- NFS 볼륨에 대한 Bottlerocket 지원이 추가되었습니다.
- Google Cloud NetApp 볼륨에 대한 기술 사전 공개 지원 추가

쿠버네티스

- Kubernetes 1.30에 대한 지원 추가
- 시작 시 좀비 마운트 및 잔여 추적 파일을 정리하는 Trident DemonSet의 기능이 추가되었습니다("문제 #883").
- LUKS 볼륨을 동적으로 가져오기 위한 PVC 주석 추가 `trident.netapp.io/luksEncryption`"문제 #849"().
- ANF 드라이버에 토폴로지 인식이 추가되었습니다.
- Windows Server 2022 노드에 대한 지원이 추가되었습니다.

수정

- 오래된 트랜잭션으로 인한 Trident 설치 실패 해결
- Kubernetes()의 경고 메시지를 무시하도록 Tridentctl을 수정했습니다"문제 #892".
- Trident 컨트롤러 우선 순위가 ("문제 #887")로 `0` 변경되었습니다 `SecurityContextConstraint`.
- ONTAP 드라이버는 이제 20MiB 이하의 볼륨 크기를 허용합니다. "문제 [#885").
- ONTAP-SAN 드라이버에 대한 크기 조정 작업 중에 FlexVol 볼륨이 축소되지 않도록 Trident를 수정했습니다.
- NFS v4.1에서 ANF 볼륨 가져오기 실패 수정.

24.02의 변경 사항

향상된 기능

- 클라우드 ID에 대한 지원이 추가되었습니다.
  - ANF-Azure 워크로드 ID가 있는 AKS는 클라우드 ID로 사용됩니다.
  - FSxN-AWS IAM 역할을 가진 EKS가 클라우드 ID로 사용됩니다.
- EKS 콘솔에서 Trident를 EKS 클러스터에 추가 기능으로 설치하기 위한 지원 추가
- iSCSI 자동 복구("문제 #864")를 클릭합니다.
- AWS IAM 및 SecretsManager와의 통합을 지원하고 Trident에서 백업을 통해 FSx 볼륨을 삭제할 수 있도록 Amazon FSx Personality를 ONTAP 드라이버에"문제 #453" 추가했습니다.

쿠버네티스

- Kubernetes 1.29에 대한 지원 추가

수정

- ACP가 활성화되지 않은 경우 ACP 경고 메시지 수정("문제 #866")를 클릭합니다.
- 클론이 스냅샷과 연결되어 있을 때 ONTAP 드라이버에 대한 스냅샷 삭제 중에 클론 분할을 수행하기 전에 10초 지연이 추가되었습니다.

사용 중단

- 다중 플랫폼 이미지 매니페스트에서 인토토 증명 프레임워크가 제거되었습니다.

23.10의 변경 사항

수정

- 새 요청된 크기가 ONTAP-NAS 및 ONTAP-NAS-flexgroup 스토리지 드라이버("문제 #834")를 클릭합니다.
- ONTAP-NAS 및 ONTAP-NAS-flexgroup 스토리지 드라이버("문제 #722")를 클릭합니다.
- ONTAP-NAS-Economy의 FlexVol 이름 변환 고정.
- 노드가 재부팅될 때 Windows 노드에서 Trident 초기화 문제가 해결되었습니다.

향상된 기능

쿠버네티스

Kubernetes 1.28에 대한 지원 추가

트라이던트

- Azure-NetApp-files 스토리지 드라이버와 함께 AMI(Azure Managed Identity)의 사용 지원 추가
- ONTAP-SAN 드라이버용 NVMe over TCP 지원 추가
- 사용자가 백엔드를 일시 중단 상태로 설정할 때 볼륨 프로비저닝을 일시 중지하는 기능이 추가되었습니다("문제 #558")를 클릭합니다.

23.07.1의 변경 사항

- Kubernetes: * 다운타임 없는 업그레이드를 지원하기 위한 데몬 세트 삭제 수정("문제 #740")를 클릭합니다.

23.07의 변경 사항

수정

쿠버네티스

- 종료 상태로 고착된 이전 Pod를 무시하도록 Trident 업그레이드를 수정했습니다("문제 #740")를 클릭합니다.
- "transient-trident-version-pod" 정의에 공차 추가("문제 #795")를 클릭합니다.

트라이던트

- 노드 스테이징 작업 중에 고스트 iSCSI 디바이스를 식별하고 수정하기 위해 LUN 속성을 가져올 때 LUN 일련

번호를 쿼리하도록 하는 고정 ONTAPI(ZAPI) 요청

- 스토리지 드라이버 코드의 오류 처리 수정("문제 #816")를 클릭합니다.
- use-rest=true인 ONTAP 드라이버를 사용할 때 할당량 크기 조정이 수정되었습니다.
- ONTAP-SAN-Economy에서 LUN 클론 생성 수정
- 에서 게시 정보 필드를 되돌립니다 `rawDevicePath` 를 선택합니다 `devicePath`; 채우기 및 복구를 위한 로직 추가(일부 경우) `devicePath` 필드에 입력합니다.

향상된 기능

쿠버네티스

- 사전 프로비저닝된 스냅샷 가져오기 지원이 추가되었습니다.
- Linux 사용 권한 최소화("문제 #817")를 클릭합니다.

트라이던트

- "온라인" 볼륨 및 스냅숏에 대한 상태 필드를 더 이상 보고하지 않습니다.
- ONTAP 백엔드가 오프라인 상태인 경우 백엔드 상태를 업데이트합니다("문제 #801", "543번")를 클릭합니다.
- LUN 일련 번호는 controllerVolumePublish 워크플로 중에 항상 검색되어 게시됩니다.
- iSCSI 다중 경로 장치의 일련 번호 및 크기를 확인하기 위한 추가 로직이 추가되었습니다.
- 올바른 다중 경로 장치가 스테이징되지 않도록 iSCSI 볼륨에 대한 추가 확인

실험 향상

ONTAP-SAN 드라이버용 NVMe over TCP에 대한 기술 미리 보기 지원 추가

문서화

많은 조직 및 서식 향상이 이루어졌습니다.

사용 중단

쿠버네티스

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

23.04의 변경 사항

ONTAP-SAN * 볼륨의 강제 볼륨 분리 기능은 비우아한 노드 종료 기능 게이트가 활성화된 Kubernetes 버전에서만 지원됩니다. 설치 시 를 사용하여 강제 분리를 활성화해야 합니다 `--enable-force-detach` Trident 설치 관리자 플래그입니다.

수정

- SPEC에 지정된 경우 설치에 IPv6 localhost를 사용하도록 고정 Trident Operator가 수정되었습니다.
- 번들 권한("문제 #799")를 클릭합니다.
- rwx 모드에서 여러 노드에 원시 블록 볼륨을 연결하는 문제 해결
- SMB 볼륨에 대한 FlexGroup 클론 복제 지원 및 볼륨 가져오기 수정
- Trident 컨트롤러가 즉시 종료되지 않는 문제 해결("문제 #811")를 클릭합니다.
- ONTAP-SAN- * 드라이버를 사용하여 프로비저닝된 지정된 LUN과 관련된 igroup의 모든 이름을 나열하는 수정 사항이 추가되었습니다.
- 외부 프로세스가 완료될 때까지 실행되도록 하는 수정 사항이 추가되었습니다.
- s390 아키텍처("문제 #537")를 클릭합니다.
- 볼륨 마운트 작업 중 잘못된 로깅 수준 수정("문제 #781")를 클릭합니다.
- 잠재적 유형 어설션 오류("문제 # 802")를 클릭합니다.

향상된 기능

- 쿠버네티스:
  - Kubernetes 1.27에 대한 지원이 추가되었습니다.
  - LUKS 볼륨 가져오기에 대한 지원이 추가되었습니다.
  - ReadWriteOncePod PVC 액세스 모드에 대한 지원이 추가되었습니다.
  - 비우아한 노드 종료 시나리오 중에 ONTAP-SAN- * 볼륨에 대한 강제 분리 지원 추가.
  - 이제 모든 ONTAP-SAN- * 볼륨에 노드당 Igroup이 사용됩니다. LUN은 igroup에 매핑되며 해당 노드에 적극적으로 게시되므로 보안 상태가 향상됩니다. Trident가 활성 워크로드(에 영향을 주지 않고 안전하게 수행할 수 있다고 판단하면 기존 볼륨이 새로운 igroup 체계로 기회적으로 전환됩니다"문제 #758")를 클릭합니다.
  - ONTAP-SAN- * 백엔드에서 사용하지 않는 Trident 관리 igroup을 정리하여 Trident 보안을 개선했습니다.
- ONTAP-NAS-이코노미 및 ONTAP-NAS-Flexgroup 스토리지 드라이버에 Amazon FSx를 포함한 SMB 볼륨 지원을 추가했습니다.
- ONTAP-NAS, ONTAP-NAS-이코노미 및 ONTAP-NAS-Flexgroup 스토리지 드라이버와 SMB 공유에 대한 지원을 추가했습니다.
- arm64 노드("문제 #732")를 클릭합니다.
- API 서버를 먼저 비활성화하여 Trident 종료 절차 개선("문제 #811")를 클릭합니다.
- Makefile에 Windows 및 arm64 호스트에 대한 교차 플랫폼 빌드 지원 추가; build.md 참조.

사용 중단

- Kubernetes:** ONTAP-SAN 및 ONTAP-SAN-이코노미 드라이버를 구성할 때 백엔드 범위의 igroup이 더 이상 생성되지 않습니다("문제 #758")를 클릭합니다.

23.01.1의 변경 사항

수정

- SPEC에 지정된 경우 설치에 IPv6 localhost를 사용하도록 고정 Trident Operator가 수정되었습니다.
- 번들 권한과 동기화되는 고정 Trident 운영자 클러스터 역할 권한 "문제 #799".
- 외부 프로세스가 완료될 때까지 실행되도록 하는 수정 사항이 추가되었습니다.
- rwx 모드에서 여러 노드에 원시 블록 볼륨을 연결하는 문제 해결
- SMB 볼륨에 대한 FlexGroup 클론 복제 지원 및 볼륨 가져오기 수정

23.01의 변경 사항

> 이제 Trident에서 Kubernetes 1.27이 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.

수정

- Kubernetes: Helm("문제 #783, #794")를 클릭합니다.

향상된 기능

쿠버네티스

- Kubernetes 1.26에 대한 지원이 추가되었습니다.
- 전체 Trident RBAC 리소스 활용률 향상("문제 #757")를 클릭합니다.
- 호스트 노드에서 손상되거나 유효하지 않은 iSCSI 세션을 감지하고 수정하는 자동화 기능 추가
- LUKS 암호화 볼륨 확장을 위한 지원이 추가되었습니다.
- Kubernetes: LUKS 암호화 볼륨에 대한 자격 증명 회전 지원 추가.

트라이던트

- Amazon FSx for NetApp ONTAP를 통한 SMB 볼륨에 대한 지원을 ONTAP-NAS 스토리지 드라이버에 추가했습니다.
- SMB 볼륨을 사용할 때 NTFS 권한에 대한 지원이 추가되었습니다.
- CVS 서비스 수준이 있는 GCP 볼륨에 대한 스토리지 풀에 대한 지원이 추가되었습니다.
- ONTAP-NAS-flexgroup 스토리지 드라이버를 사용하여 FlexGroups를 생성할 때 flexgroupAggregateList의 선택적 사용에 대한 지원이 추가되었습니다.
- 여러 개의 FlexVol 볼륨을 관리할 때 ONTAP-NAS 경제적인 스토리지 드라이버의 성능이 향상되었습니다
- 모든 ONTAP NAS 스토리지 드라이버에 대해 데이터 LIF 업데이트를 사용하도록 설정했습니다.
- 호스트 노드 OS를 반영하도록 Trident 배포 및 DemonSet 명명 규칙을 업데이트했습니다.

사용 중단

- Kubernetes: 지원되는 최소 Kubernetes를 1.21로 업데이트했습니다.

- 또는 `ontap-san-economy` 드라이버를 구성할 때 DataLIF를 더 이상 지정하지 `ontap-san` 않아야 합니다.

22.10의 변경 사항

Trident 22.10으로 업그레이드하기 전에 다음 중요 정보를 읽어야 합니다**.**



** Trident 22.10 **에 대한 중요 정보입니다

- Kubernetes 1.25가 이제 Trident에서 지원됩니다. Kubernetes 1.25로 업그레이드하기 전에 Trident를 22.10으로 업그레이드해야 합니다.
- Trident은 이제 SAN 환경에서 다중 경로 구성을 엄격하게 적용하며 multipath.conf 파일에서 권장 값은 `find_multipaths: no` 입니다.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

수정

- 을 사용하여 생성된 ONTAP 백엔드에 대한 특정 문제를 해결했습니다 `credentials` 22.07.0 업그레이드 중 온라인 상태로 전환되지 않는 필드("문제 #759")를 클릭합니다.
- Docker: 일부 환경에서 Docker 볼륨 플러그인을 시작하지 못하는 문제 해결("문제 #548" 및 "문제 # 760")를 클릭합니다.
- 보고 노드에 속하는 데이터 LIF의 하위 집합만 게시되도록 ONTAP SAN 백엔드와 관련된 SLM 문제를 수정했습니다.
- 볼륨을 연결할 때 iSCSI LUN에 대한 불필요한 검사가 발생하는 성능 문제를 해결했습니다.
- Trident iSCSI 워크플로우 내에서 세분화된 재시도를 제거하여 빠르게 실패하고 외부 재시도 간격을 줄입니다.
- 해당 다중 경로 장치가 이미 플러시되었을 때 iSCSI 장치를 플러싱할 때 오류가 반환되는 문제를 해결했습니다.

향상된 기능

- 쿠버네티스:
  - Kubernetes 1.25에 대한 지원 추가 Kubernetes 1.25로 업그레이드하기 전에 Trident를 22.10으로 업그레이드해야 합니다.
  - Trident Deployment 및 DemonSet에 대해 별도의 ServiceAccount, ClusterRole 및 ClusterRoleBinding을 추가하여 이후의 사용 권한 개선을 허용합니다.
  - 에 대한 지원이 추가되었습니다 "네임스페이스 간 볼륨 공유".
- 모든 Trident `ontap-*` 이제 스토리지 드라이버가 ONTAP REST API에서 작동합니다.
- 새로운 운영자 YAML 추가 (`bundle_post_1_25.yaml`)를 사용하지 않을 경우 `PodSecurityPolicy` Kubernetes 1.25를 지원합니다.
- 추가되었습니다 "LUKS 암호화 볼륨 지원" 용 `ontap-san` 및 `ontap-san-economy` 스토리지 드라이버.
- Windows Server 2019 노드에 대한 지원이 추가되었습니다.
- 추가되었습니다 "Windows 노드에서 SMB 볼륨 지원" 를 통해 `azure-netapp-files` 스토리지 드라이버.
- 이제 ONTAP 드라이버에 대한 자동 MetroCluster 전환 감지 기능을 사용할 수 있습니다.

사용 중단

- **Kubernetes:** 최소 지원 Kubernetes를 1.20으로 업데이트했습니다.
- ADS(Astra Data Store) 드라이버를 제거했습니다.
- 에 대한 지원이 제거되었습니다 `yes` 및 `smart` 에 대한 옵션 `find_multipaths` iSCSI에 대한 작업자 노드 경로 다중화를 구성하는 경우

22.07 변경

수정

- Kubernetes**
  - Hrom 또는 Trident 연산자를 사용하여 Trident를 구성할 때 노드 선택기에 대한 부울 및 숫자 값을 처리하는 문제가 해결되었습니다. ("GitHub 문제 #700")
  - CHAP가 아닌 경로에서 발생하는 오류를 처리하는 문제를 수정함으로써 kubelet이 실패한 경우 다시 시도합니다. "GitHub 문제 #736")

향상된 기능

- k8s.gcr.io에서 registry.k8s.io로 CSI 이미지의 기본 레지스트리로 전환합니다
- ONTAP-SAN 볼륨은 이제 노드별 igroup을 사용하며 해당 노드에 능동적으로 게시되는 LUN만 igroup에 매핑하여 보안 상태를 향상합니다. Trident가 활성 워크로드에 영향을 주지 않고 안전하게 수행할 수 있다고 판단하면 기존 볼륨이 새로운 igroup 체계로 자동으로 전환됩니다.
- Trident 설치에 리소스 할당량을 포함함으로써 PriorityClass 소비가 기본적으로 제한될 때 Trident DemonSet이 예약되도록 합니다.
- Azure NetApp Files 드라이버에 네트워크 기능에 대한 지원이 추가되었습니다. ("GitHub 문제 #717")
- ONTAP 드라이버에 기술 미리 보기 자동 MetroCluster 전환 감지 기능이 추가되었습니다. ("GitHub 문제 #228")

사용 중단

- **Kubernetes:** 최소 지원 Kubernetes를 1.19로 업데이트했습니다.
- 백엔드 구성은 더 이상 단일 구성에서 여러 인증 유형을 사용할 수 없습니다.

제거

- AWS CVS 드라이버(22.04 이후 더 이상 사용되지 않음)가 제거되었습니다.
- 쿠버네티스
  - 노드 포드에 불필요한 SYS_ADMIN 기능이 제거되었습니다.
  - 노드 준비 작업을 간단한 호스트 정보로 줄이고 활성 서비스 검색을 통해 작업 노드에서 NFS/iSCSI 서비스를 사용할 수 있다는 최선의 확인 작업을 수행할 수 있습니다.

문서화

설치 시 Trident에서 사용할 수 있는 권한을 자세히 설명하는 새"POD 보안 표준"(PSS) 섹션이 추가되었습니다.

22.04의 변경

NetApp은 제품과 서비스를 지속적으로 개선 및 개선하고 있습니다. Trident의 최신 기능은 다음과 같습니다. 이전 릴리스는 을 "이전 버전의 문서"참조하십시오.

> 이전 Trident 릴리스에서 업그레이드하고 Azure NetApp Files를 사용하는 경우 이제 "location" config 매개 변수는 필수 싱글톤 필드입니다.

수정

- iSCSI 이니시에이터 이름의 구문 분석 기능이 향상되었습니다. ("GitHub 문제 #681")
- CSI 스토리지 클래스 매개 변수가 허용되지 않는 문제를 해결했습니다. ("GitHub 문제 #598")
- Trident CRD에서 중복 키 선언을 수정했습니다. ("GitHub 문제 #671")
- 부정확한 CSI 스냅샷 로그를 수정했습니다. ("GitHub 문제 #629"))
- 삭제된 노드에서 볼륨 게시를 취소하는 문제 해결 ("GitHub 문제 #691")
- 블록 디바이스에서 파일 시스템 불일치를 처리하는 기능이 추가되었습니다. ("GitHub 문제 #656")
- 설치 중 'imageRegistry' 플래그를 설정할 때 자동 지원 이미지를 가져오는 문제가 해결되었습니다. ("GitHub 문제 #715")
- Azure NetApp Files 드라이버가 여러 내보내기 규칙을 사용하여 볼륨을 복제하지 못하는 문제가 해결되었습니다.

향상된 기능

- 이제 Trident의 보안 끝점에 대한 인바운드 연결에는 TLS 1.3 이상이 필요합니다. ("GitHub 문제 #698")
- 이제 Trident는 보안 엔드포인트의 응답에 HSTS 헤더를 추가합니다.
- 이제 Trident는 Azure NetApp Files UNIX 사용 권한 기능을 자동으로 활성화하려고 시도합니다.
- * Kubernetes *: Trident가 이제 시스템 노드 크리티컬 우선 순위 클래스에서 실행됩니다. ("GitHub 문제 #694")

제거

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

22.01.1의 변경 사항

수정

- 삭제된 노드에서 볼륨 게시를 취소하는 문제 해결 ("GitHub 문제 #691")
- ONTAP API 응답에서 공간 집계에 대한 nil 필드에 액세스할 때 패닉이 수정되었습니다.

22.01.0의 변경 사항

수정

- * Kubernetes: * 대규모 클러스터의 노드 등록 백오프 재시도 시간을 늘립니다.
- 동일한 이름의 여러 리소스가 Azure-NetApp-files 드라이버를 혼동할 수 있는 문제 해결

- ONTAP SAN IPv6 DataLIF는 이제 대괄호로 지정된 경우 작동합니다.
- 이미 가져온 볼륨을 가져오려고 하면 PVC가 보류 상태로 남겨둔 EOF가 반환되는 문제가 해결되었습니다. ("GitHub 문제 #489")
- SolidFire 볼륨에 32개가 넘는 스냅샷이 생성될 때 Trident 성능이 저하되는 문제가 해결되었습니다.
- SHA-1을 SSL 인증서 생성에서 SHA-256으로 교체했습니다.
- Azure NetApp Files 드라이버를 수정하여 중복된 리소스 이름을 허용하고 단일 위치로 작업을 제한했습니다.
- Azure NetApp Files 드라이버를 수정하여 중복된 리소스 이름을 허용하고 단일 위치로 작업을 제한했습니다.

향상된 기능

- Kubernetes의 향상된 기능:
  - Kubernetes 1.23 지원 추가.
  - Trident Operator 또는 Hrom을 통해 설치된 Trident Pod에 대한 예약 옵션을 추가합니다. ("GitHub 문제 #651")
- GCP 드라이버에서 지역 간 볼륨을 허용합니다. ("GitHub 문제 #633")
- Azure NetApp Files 볼륨에 'unixPermissions' 옵션 지원이 추가되었습니다. ("GitHub 문제 #666")

사용 중단

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

21.10.1 변경



v21.10.0 릴리즈에는 노드를 제거한 다음 Kubernetes 클러스터에 다시 추가할 때 Trident 컨트롤러를 CrashLoopBackOff 상태로 전환할 수 있는 문제가 있습니다. 이 문제는 v21.10.1(GitHub 문제 669)에서 해결되었습니다.

수정

- GCP CVS 백엔드에서 볼륨을 가져올 때 잠재적인 경쟁 조건이 수정되어 가져오지 못했습니다.
- 노드를 제거할 때 Trident 컨트롤러를 CrashLoopBackOff 상태로 전환할 수 있는 문제를 해결한 다음 Kubernetes 클러스터(GitHub 문제 669)에 다시 추가되었습니다.
- SVM 이름이 지정되지 않은 경우 SVM이 더 이상 검색되지 않는 문제 해결(GitHub 문제 612)

21.10.0 변경

수정

- XFS 볼륨의 클론을 소스 볼륨과 동일한 노드에 마운트할 수 없는 문제(GitHub 문제 514)가 해결되었습니다.
- Trident에서 종료 시 심각한 오류를 기록하는 문제가 해결되었습니다(GitHub 문제 597).
- Kubernetes 관련 수정 사항:
  - ONTAP-NAS와 ONTAP-NAS-Flexgroup 드라이버(GitHub 645호)를 사용하여 스냅샷을 생성할 때 볼륨의 사용된 공간을 최소 restoreSize로 반환합니다.

- 볼륨 크기 조정(GitHub 문제 560) 후 "파일 시스템 확장 실패" 오류가 기록된 문제 해결
- 포드가 종단 상태로 고착되는 문제 해결(GitHub 문제 572)
- ONTAP-SAN-E경제적인 FlexVol에 스냅샷 LUN이 포함될 수 있는 사례 해결(GitHub 문제 533)
- 다른 이미지의 사용자 지정 YAML 설치 프로그램 문제 해결(GitHub 문제 613)
- 스냅샷 크기 계산 수정(GitHub 문제 611)
- 모든 Trident 설치 관리자가 일반 Kubernetes를 OpenShift로 식별할 수 있는 문제가 해결되었습니다(GitHub 문제 639).
- Kubernetes API 서버에 연결할 수 없는 경우 조정을 중지하도록 Trident 연산자를 수정했습니다(GitHub 문제 599).

향상된 기능

- GCP-CVS 성능 볼륨에 대한 "unixPermissions" 옵션에 대한 지원이 추가되었습니다.
- 600GiB~1TiB 범위의 GCP에서 확장성 최적화 CVS 볼륨 지원 추가
- Kubernetes 관련 개선사항:
  - Kubernetes 1.22에 대한 지원이 추가되었습니다.
  - Trident 운영자 및 제어 차트를 Kubernetes 1.22(GitHub 문제 628)와 함께 사용할 수 있도록 했습니다.
  - tridentctl 이미지 명령어(GitHub 문제 570)에 운용자 이미지 추가

실험적인 개선

- 'ONTAP-SAN' 드라이버에 볼륨 복제 지원이 추가되었습니다.
- ONTAP-NAS-Flexgroup, ONTAP-SAN, ONTAP-NAS-이코노미 드라이버에 대한 * 기술 미리보기 * REST 지원이 추가되었습니다.

알려진 문제

알려진 문제점은 제품을 성공적으로 사용하지 못하게 만들 수 있는 문제를 식별합니다.

- Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드할 `true` `helm upgrade` 경우 클러스터를 업그레이드하기 전에 value.yaml을 명령으로 설정하거나 `--set excludePodSecurityPolicy=true` 명령에 추가해야 `excludePodSecurityPolicy` 합니다.
- 이제 Trident는 해당 StorageClass에 지정되지(`fsType=""` 않은 볼륨에 대해 공백으로 `fsType` 표시합니다. `fsType` Kubernetes 1.17 이상으로 작업하는 경우 Trident에서는 NFS 볼륨에 공백 제공을 `fsType` 지원합니다. iSCSI 볼륨의 경우 보안 컨텍스트를 사용하여 를 적용할 때 StorageClass에서 `fsGroup` 을 설정해야 `fsType` 합니다.
- 여러 Trident 인스턴스에 걸쳐 백엔드를 사용할 경우 각 백엔드 구성 파일의 ONTAP 백엔드에 대해 다른 값을 사용하거나 SolidFire 백엔드에 대해 다른 값을 `TenantName` 사용해야 `storagePrefix` 합니다. Trident은 다른 Trident 인스턴스에서 생성한 볼륨을 감지할 수 없습니다. Trident는 볼륨 생성을 유용한 작업으로 취급하므로 ONTAP 또는 SolidFire 백엔드에서 기존 볼륨을 생성하려고 시도하면 성공합니다. 또는 `TenantName` 이 차이가 없으면 `storagePrefix` 동일한 백엔드에서 생성된 볼륨에 대한 이름 충돌이 발생할 수 있습니다.
- Trident(또는 Trident 연산자 사용)를 설치하고 를 사용하여 `tridentctl` Trident를 관리하는 경우 `tridentctl` 환경 변수가 설정되어 있는지 확인해야 `KUBECONFIG` 합니다. 이는 작업할 Kubernetes 클러스터를 나타내는 데 `tridentctl` 필요합니다. 여러 Kubernetes 환경에서 작업할 때는 파일을 정확하게 소싱해야 `KUBECONFIG`

합니다.

- iSCSI PVS에 대해 온라인 공간 재확보를 수행하려면 작업자 노드의 기본 OS에 볼륨에 마운트 옵션을 전달해야 할 수 있습니다. 이는 가 필요한 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS) 인스턴스의 `discard` "마운트 옵션" 경우에도 마찬가지입니다. 온라인 블록 폐기를 지원하기 위해 discard mountoption이 사용자의 에 포함되어 [`StorageClass` 있는지 확인하십시오.
- Kubernetes 클러스터당 Trident 인스턴스가 두 개 이상인 경우 Trident는 다른 인스턴스와 통신할 수 없으며 새로 생성한 다른 볼륨을 찾을 수 없습니다. 따라서 클러스터 내에서 인스턴스가 두 개 이상 실행될 경우 예기치 못한 동작이 발생합니다. Kubernetes 클러스터당 Trident 인스턴스는 하나만 있어야 합니다.
- Trident이 오프라인일 때 Trident 기반 오브젝트가 Kubernetes에서 삭제되는 경우 `StorageClass`, Trident은 다시 온라인 상태가 될 때 데이터베이스에서 해당 스토리지 클래스를 제거하지 않습니다. 또는 REST API를 사용하여 이러한 스토리지 클래스를 `tridentctl` 삭제해야 합니다.
- 사용자가 해당 PVC를 삭제하기 전에 Trident에서 제공하는 PV를 삭제할 경우 Trident는 백업 볼륨을 자동으로 삭제하지 않습니다. 또는 REST API를 통해 볼륨을 제거해야 `tridentctl` 합니다.
- FlexGroup은 애그리게이트 세트가 각 프로비저닝 요청에서 고유하지 않으면 한 번에 둘 이상의 ONTAP를 동시에 프로비저닝할 수 없습니다.
- IPv6를 통한 Trident를 사용할 때는 백엔드의 정의에 대괄호로 및 `dataLIF` 을 지정해야 `managementLIF` 합니다. `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`예를 들어,

> ONTAP SAN 백엔드에는 을 지정할 수 `dataLIF` 없습니다. Trident은 사용 가능한 모든 iSCSI LIF를 검색하여 다중 경로 세션을 설정합니다.

- 를 사용하는 경우 `solidfire-san` OpenShift 4.5를 사용하는 드라이버 기본 작업자 노드가 CHAP 인증 알고리즘으로 MD5를 사용하는지 확인합니다. 보안 FIPS 호환 CHAP 알고리즘 SHA1, SHA-256 및 SHA3-256은 Element 12.7에서 사용할 수 있습니다.

자세한 내용을 확인하십시오

- "GitTrident 허브"
- "Trident 블로그"

이전 버전의 문서

Trident 26.10을 실행 중이 아닌 경우 이전 릴리스에 대한 문서는 "Trident 지원 라이프사이클"을 기반으로 제공됩니다.

- "Trident 26.06"
- "Trident 26.02"
- "Trident 25.10"
- "Trident 25.06"
- "Trident 25.02 을 참조하십시오"
- "Trident 24.10 을 참조하십시오"
- "Trident 24.06 을 참조하십시오"
- "Trident 24.02 를 참조하십시오"

- "Trident 23.10 을 참조하십시오"
- "Trident 2007년 3월"
- "Trident 23.04 를 참조하십시오"
- "Trident 23.01 을 참조하십시오"

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.
- Trident 25.10 이상에서는 iSCSI 및 NVMe/TCP 외에도 FC를 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

- Linux Unified Key Setup (LUKS) 암호화
- SnapMirror 볼륨 복제
- 총 사용량 제한
- 공간 예약 모드
- 스냅샷 수

- 계층화

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

알려진 문제

알려진 문제점은 이 제품 릴리스를 성공적으로 사용하지 못하게 만들 수 있는 문제를 식별합니다.

현재 릴리즈에는 다음과 같은 알려진 문제가 영향을 줍니다.

Trident 22.10 또는 이전 버전에서 업그레이드한 후에도 볼륨은 연결된 상태로 유지됩니다.

이 문제는 볼륨이 연결된 상태에서 Trident 22.10 또는 이전 버전에서 Trident 26.10으로 직접 업그레이드하는 고객에게 영향을 미칩니다. Trident 23.01 이상 버전에서의 업그레이드는 영향을 받지 않습니다.

업그레이드 후 영향을 받는 볼륨이 분리되지 않을 수 있습니다. ReadWriteOnce(RWO) 볼륨을 사용하는 Pod는 다른 노드로 이동할 수 없으며 다중 연결 오류가 발생할 수 있습니다. PersistentVolume(PV)을 삭제하는 작업도 중단될 수 있습니다.

이 문제를 해결하려면 오래된 볼륨 연결이 있는 노드에서 해당 볼륨의 `globalmount` 디렉토리에서 남아 있는 `volumePublishInfo.json` 파일을 삭제하십시오.

VolumeSnapshots이 ReadyToUse 상태에 도달하지 못했습니다

> `csi-snapshotter` 사이드카에 알려진 문제가 있습니다. 모든 `Kubernetes` 버전에서 `VolumeGroupSnapshots` `v1beta1`는 `VolumeSnapshots`가 `ReadyToUse` 상태에 도달하는 것을 방지합니다.

두 가지 해결 방법이 있습니다.

1. VolumeGroupSnapshots를 비활성화하려면 VolumeGroupSnapshots CRD를 삭제한 다음 Trident를 다시 설치합니다.
2. VolumeGroupSnapshots `v1beta2` 및 `snapshot-controller` 버전 8.4.0 이상을 설치한 다음 Trident를 다시 설치합니다. VolumeGroupSnapshots는 Kubernetes 버전 v1.34 이전 버전에서는 작동하지 않습니다.

대용량 파일의 임시 백업을 복원하지 못할 수 있습니다

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

시작하십시오

Trident에 대해 자세히 알아보십시오

Trident에 대해 자세히 알아보십시오

Trident는 NetApp에서 관리하며 완벽한 지원이 제공되는 오픈 소스 프로젝트입니다. CSI(Container Storage Interface)와 같은 업계 표준 인터페이스를 사용하여 컨테이너식 애플리케이션의 지속성 요구 사항을 충족하도록 설계되었습니다.

Trident란 무엇입니까?

NetApp Trident는 퍼블릭 클라우드 또는 온프레미스 환경에서 모든 인기 있는 NetApp 스토리지 플랫폼의 스토리지 리소스 소비 및 관리를 가능하게 하며, 온프레미스 ONTAP 클러스터(AFF, FAS, ASA), ONTAP Select, Cloud Volumes ONTAP, Element 소프트웨어(NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP, Google Cloud NetApp Volumes를 포함합니다.

Trident는 기본적으로 와 통합되는 CSI(컨테이너 스토리지 인터페이스) 호환 동적 스토리지 오케스트레이터입니다. "쿠버네티스" Trident는 클러스터의 각 작업자 노드에서 단일 컨트롤러 Pod와 노드 Pod로 실행됩니다. 자세한 내용은 을 "Trident 아키텍처" 참조하십시오.

Trident는 또한 NetApp 스토리지 플랫폼을 위한 Docker 에코시스템과의 직접 통합을 제공합니다. NetApp Docker 볼륨 플러그인(nDVP)은 스토리지 플랫폼에서 Docker 호스트로 스토리지 리소스를 프로비저닝 및 관리할 수 있도록 지원합니다. 자세한 내용은 을 "Docker용 Trident를 구축하십시오" 참조하십시오.

Kubernetes를 처음으로 사용하는 경우라면 에 익숙해야 합니다 "Kubernetes 개념 및 툴".

지원되는 Kubernetes 플랫폼

Trident는 다양한 Kubernetes 배포판 및 플랫폼에서 지원됩니다.

지원되는 플랫폼은 다음과 같습니다. * 업스트림 Kubernetes * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

Kubernetes와 NetApp 제품의 통합

NetApp 스토리지 제품 포트폴리오는 Kubernetes 클러스터의 다양한 측면과 통합되어 Kubernetes 구축의 기능, 기능, 성능 및 가용성을 향상하는 고급 데이터 관리 기능을 제공합니다.

▼ **NetApp ONTAP용 Amazon FSx**

"NetApp ONTAP용 Amazon FSx" 는 NetApp ONTAP 스토리지 운영 체제에 기반한 파일 시스템을 시작하고 실행할 수 있는 완전 관리형 AWS 서비스입니다.

▼ **Azure NetApp Files**

"Azure NetApp Files" NetApp에서 제공하는 엔터프라이즈급 Azure 파일 공유 서비스입니다. Azure에서 기본적으로 가장 까다로운 파일 기반 워크로드를 실행하고 NetApp에서 기대하는 성능 및 강력한 데이터 관리를 제공할 수 있습니다.

▼ **Cloud Volumes ONTAP**

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

▼ **Google Cloud NetApp** 볼륨

"Google Cloud NetApp 볼륨" 는 Google Cloud에서 제공하는 완전 관리형 파일 스토리지 서비스로서, 고성능 엔터프라이즈급 파일 스토리지를 제공합니다.

▼ **Element** 소프트웨어

"요소" 스토리지 관리자가 성능을 보장하고 단순화된 스토리지 설치 공간을 활용하여 워크로드를 통합할 수 있도록 지원합니다.

▼ **NetApp HCI**

"NetApp HCI" 일상적인 작업을 자동화하고 인프라 관리자가 보다 중요한 기능에 집중할 수 있도록 하여 데이터 센터의 관리 및 확장을 단순화합니다.

Trident는 컨테이너화된 애플리케이션에 대한 스토리지 장치를 기본 NetApp HCI 스토리지 플랫폼에 직접 프로비저닝 및 관리할 수 있습니다.

▼ **NetApp ONTAP**를 참조하십시오

"NetApp ONTAP를 참조하십시오" 는 모든 애플리케이션에 고급 데이터 관리 기능을 제공하는 NetApp 멀티 프로토콜 유니파이드 스토리지 운영 체제입니다.

ONTAP 시스템은 All-Flash, 하이브리드 또는 All-HDD 구성을 가지며 온프레미스 FAS, AFA, ASA 클러스터, ONTAP Select, Cloud Volumes ONTAP 등 다양한 구축 모델을 제공합니다. Trident는 이러한 ONTAP 구축 모델을 지원합니다.

Trident 아키텍처

Trident는 클러스터의 각 작업자 노드에서 단일 컨트롤러 Pod와 노드 Pod로 실행됩니다. 노드 포드는 Trident 볼륨을 마운트하려는 호스트에서 실행되고 있어야 합니다.

컨트롤러 Pod 및 노드 포드 이해

Trident는 Kubernetes 클러스터에 하나 이상의 Trident 노드 Pod단일 시스템으로 배포되고 Trident 컨트롤러 Pod표준 Kubernetes_CSI Sidecar Containers_를 사용하여 CSI 플러그인 구축을 간소화합니다. "Kubernetes CSI Sidecar Containers의 약어입니다" Kubernetes 스토리지 커뮤니티에서 유지 관리합니다.

Kubernetes"노드 선택기"로, "관용과 오해"특정 노드 또는 기본 노드에서 실행할 Pod를 제한하는 데 사용됩니다. Trident 설치 중에 컨트롤러 및 노드 Pod의 노드 선택기 및 허용을 구성할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

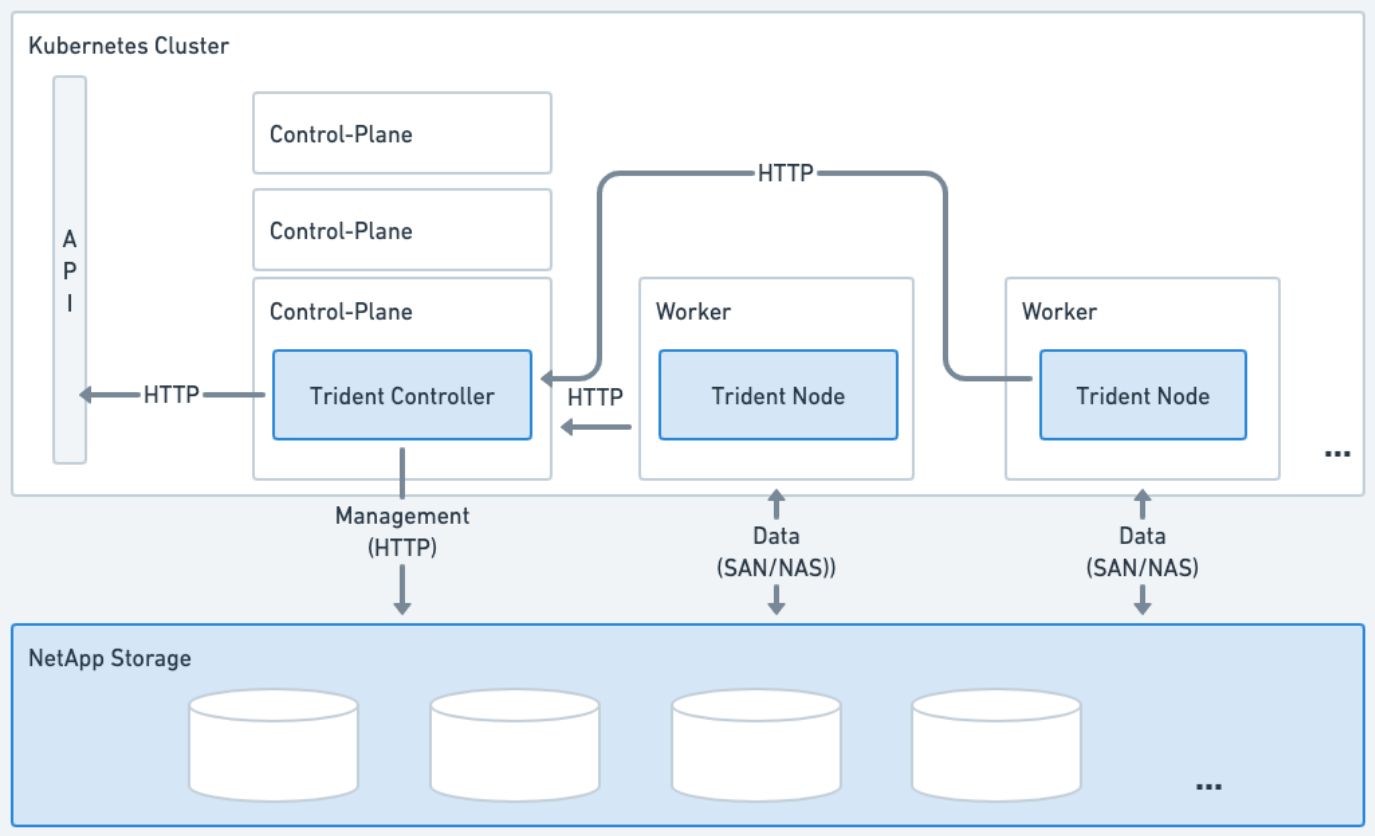


그림 **1. Kubernetes** 클러스터에 배포된 **Trident**

Trident 컨트롤러 **Pod**

Trident 컨트롤러 Pod는 CSI 컨트롤러 플러그인을 실행하는 단일 Pod입니다.

- NetApp 스토리지에서 볼륨을 프로비저닝하고 관리하는 업무를 담당합니다
- Kubernetes 배포에 의해 관리됩니다
- 설치 매개변수에 따라 컨트롤 플레인 또는 작업자 노드에서 실행할 수 있습니다.

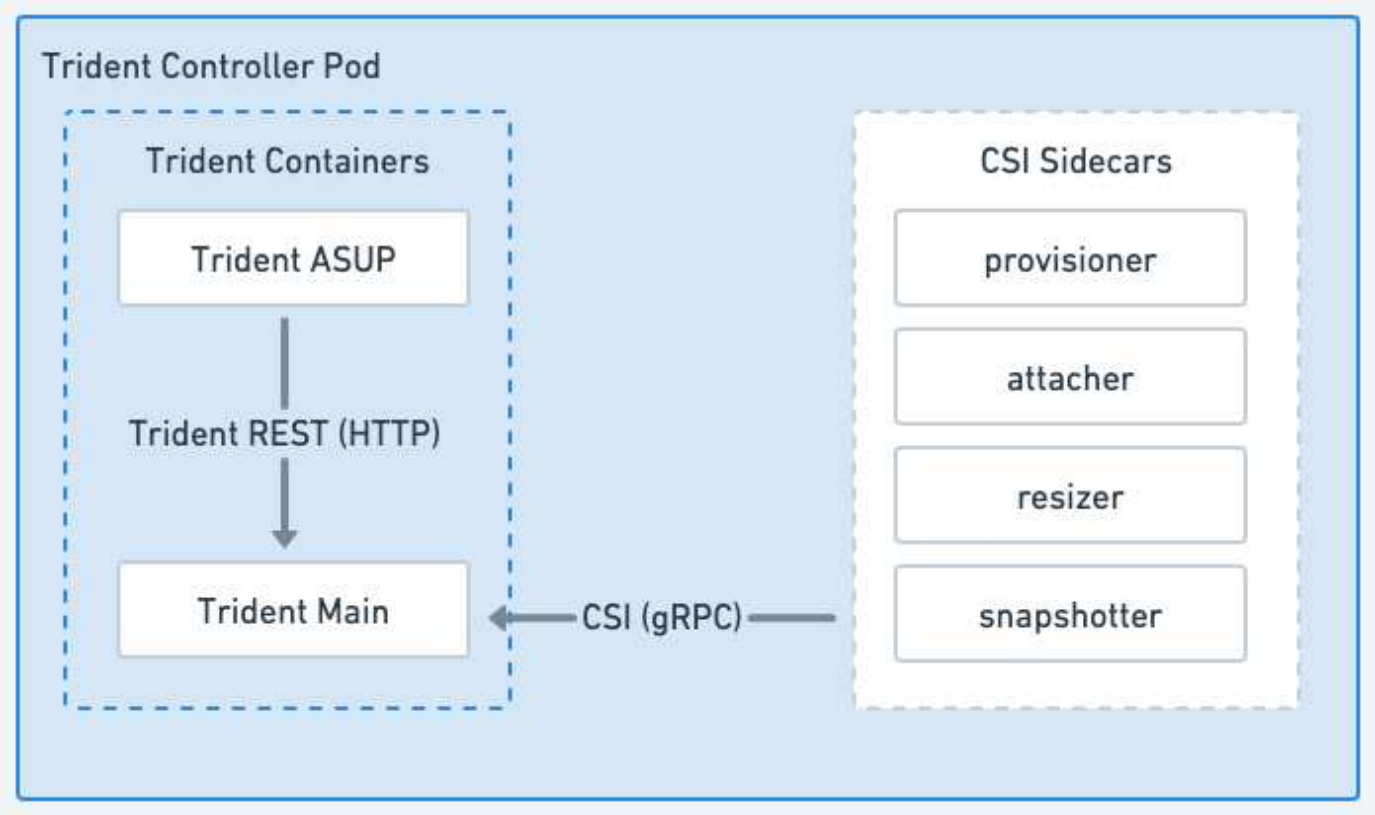


그림 **2. Trident** 컨트롤러 **Pod** 다이어그램

Trident 노드 **Pod**

Trident 노드 Pod는 CSI 노드 플러그인을 실행하는 권한 있는 Pod입니다.

- 호스트에서 실행 중인 Pod에 대한 스토리지 마운트 및 마운트 해제를 담당합니다
- Kubernetes DemonSet에서 관리합니다
- NetApp 스토리지를 마운트할 모든 노드에서 실행해야 합니다

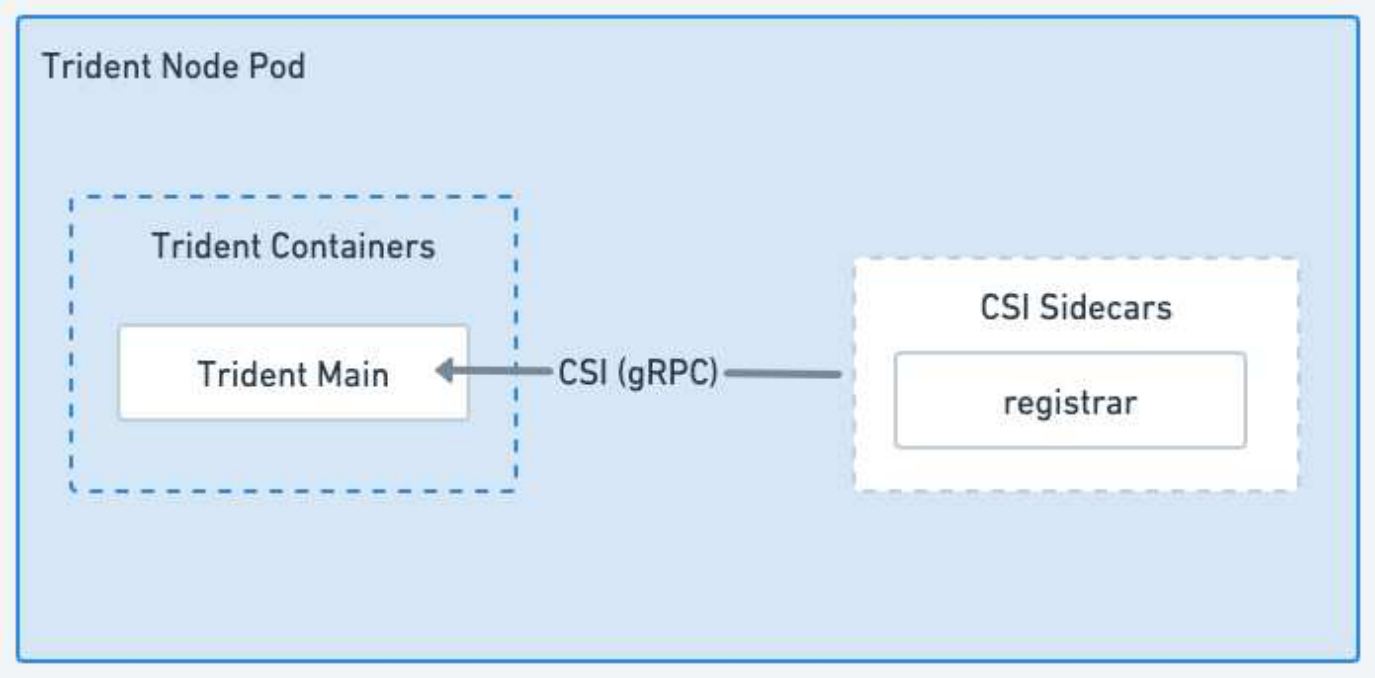


그림 **3. Trident** 노드 **Pod** 다이어그램

지원되는 **Kubernetes** 클러스터 아키텍처

Trident는 다음 Kubernetes 아키텍처에서 지원됩니다.

Kubernetes 클러스터 아키텍처	지원	기본 설치
단일 마스터, 컴퓨팅	예	예
다중 마스터, 컴퓨팅	예	예
마스터, "etcd", 컴퓨팅	예	예
마스터, 인프라, 컴퓨팅	예	예

개념

Trident 볼륨 프로비저닝 프로세스

Trident에서의 프로비저닝은 기본적으로 두 가지 단계로 구성됩니다. 첫 번째 단계에서는 스토리지 클래스를 적절한 백엔드 스토리지 풀 세트와 연결하고 용량 할당 전에 필요한 준비 작업으로 수행됩니다. 두 번째 단계에서는 볼륨 생성 자체를 포함하며 보류 중인 볼륨의 스토리지 클래스와 연결된 스토리지 풀을 선택해야 합니다.

스토리지 클래스 연결

백엔드 스토리지 풀을 스토리지 클래스와 연결하려면 스토리지 클래스의 요청된 속성과 해당 속성, `additionalStoragePools` 및 `excludeStoragePools` 목록에 따라 `storagePools` 달라집니다. 스토리지 클래스를 생성할 때 Trident는 각 백엔드에서 제공하는 특성과 풀을 스토리지 클래스에서 요청한 속성과 비교합니다. 스토리지 풀의 속성 및 이름이 요청된 속성 및 풀 이름과 모두 일치하는 경우 Trident는 해당 스토리지 클래스에 적합한 스토리지 풀 세트에 해당 스토리지 풀을 추가합니다. 또한 Trident는 목록에 나열된 모든 스토리지 풀을 해당 집합에 추가합니다. 이러한 스토리지 풀의 `additionalStoragePools` 속성이 스토리지 클래스의 요청된 속성 중 일부 또는

전부를 충족하지 않는 경우에도 마찬가지입니다. 이 목록을 사용하여 `excludeStoragePools` 스토리지 클래스에 대해 스토리지 풀을 재정의하고 사용할 수 없도록 제거해야 합니다. Trident는 새 백엔드를 추가할 때마다 유사한 프로세스를 수행하여 해당 스토리지 풀이 기존 스토리지 클래스의 스토리지 풀을 만족하는지 확인하고 제외된 것으로 표시된 스토리지 풀을 제거합니다.

볼륨 생성

그런 다음 Trident는 스토리지 클래스와 스토리지 풀 간의 연결을 사용하여 볼륨을 프로비저닝할 위치를 결정합니다. 볼륨을 생성할 때 Trident는 먼저 해당 볼륨의 스토리지 클래스에 대한 스토리지 풀 세트를 가져오고, 볼륨에 대한 프로토콜을 지정하면 Trident는 요청된 프로토콜을 제공할 수 없는 스토리지 풀을 제거합니다(예: NetApp HCI/SolidFire 백엔드는 파일 기반 볼륨을 제공할 수 없지만 ONTAP NAS 백엔드는 블록 기반 볼륨을 제공할 수 없음). Trident은 볼륨을 균등하게 분산하기 위해 결과 세트의 순서를 무작위로 추출한 다음 이 순서를 반복하여 각 스토리지 풀에 볼륨을 차례로 프로비저닝합니다. 이 오류가 1에서 성공하면 프로세스에서 발생한 모든 오류를 로깅하여 성공적으로 반환됩니다. Trident는 요청된 스토리지 클래스 및 프로토콜에 사용할 수 있는 스토리지 풀을 * 모든 * 에서 프로비저닝하지 못한 경우에만 오류 * 를 반환합니다.

Trident 볼륨 스냅샷 생성

Trident에서 드라이버에 대한 볼륨 스냅샷 생성을 어떻게 처리하는지 자세히 알아보십시오.

볼륨 스냅샷 생성에 대해 자세히 알아보십시오

- 를 위해 `ontap-nas` , `ontap-san` , 그리고 `azure-netapp-files` 드라이버, 각 영구 볼륨(PV)은 FlexVol volume 에 매핑됩니다. 결과적으로 볼륨 스냅샷은 NetApp 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁사의 스냅샷 기술보다 더 뛰어난 안정성, 확장성, 복구 가능성 및 성능을 제공합니다. 이러한 스냅샷 복사본은 복사본을 만드는 데 필요한 시간과 저장 공간 측면에서 매우 효율적입니다.
- 의 경우 `ontap-nas-flexgroup` 드라이버, 각 PV(영구 볼륨)는 FlexGroup에 매핑됩니다. 따라서 볼륨 스냅샷이 NetApp FlexGroup 스냅샷으로 생성됩니다. NetApp 스냅샷 기술은 경쟁 스냅샷 기술보다 더 높은 안정성, 확장성, 복구 가능성 및 성능을 제공합니다. 이러한 스냅샷 복사본은 생성하는 데 필요한 시간과 스토리지 공간 모두에서 매우 효율적입니다.
- 드라이버의 경우 `ontap-san-economy` PVS는 공유 FlexVol 볼륨에 생성된 LUN에 매핑됩니다. PVS의 볼륨 스냅샷은 연결된 LUN의 FlexClone을 수행하여 생성됩니다. ONTAP FlexClone 기술을 사용하면 가장 큰 데이터 세트의 복사본을 거의 즉시 생성할 수 있습니다. 복사본은 데이터 블록을 부모님과 공유하므로 메타데이터에 필요한 것만 빼고 스토리지를 사용하지 않습니다.
- '솔라이드산' 드라이버의 경우, 각 PV는 NetApp Element 소프트웨어/NetApp HCI 클러스터에서 생성된 LUN에 매핑됩니다. 볼륨 스냅샷은 기본 LUN의 요소 스냅샷으로 표시됩니다. 이러한 스냅샷은 특정 시점 복제본이며 소량의 시스템 리소스와 공간만 차지합니다.
- 및 `ontap-san` 드라이버로 작업할 때 `ontap-nas` ONTAP 스냅샷은 FlexVol의 시점 복제본이며 FlexVol 자체에서 공간을 사용합니다. 따라서 스냅샷이 생성/예약될 때 시간이 경과함에 따라 볼륨의 쓰기 가능한 공간이 줄어들 수 있습니다. 이 문제를 해결하는 간단한 방법 중 하나는 Kubernetes를 통해 크기를 조정하여 볼륨을 늘리는 것입니다. 또 다른 옵션은 더 이상 필요하지 않은 스냅샷을 삭제하는 것입니다. Kubernetes를 통해 생성된 볼륨 스냅샷이 삭제되면 Trident는 연결된 ONTAP 스냅샷을 삭제합니다. Kubernetes를 통해 생성되지 않은 ONTAP 스냅샷도 삭제할 수 있습니다.

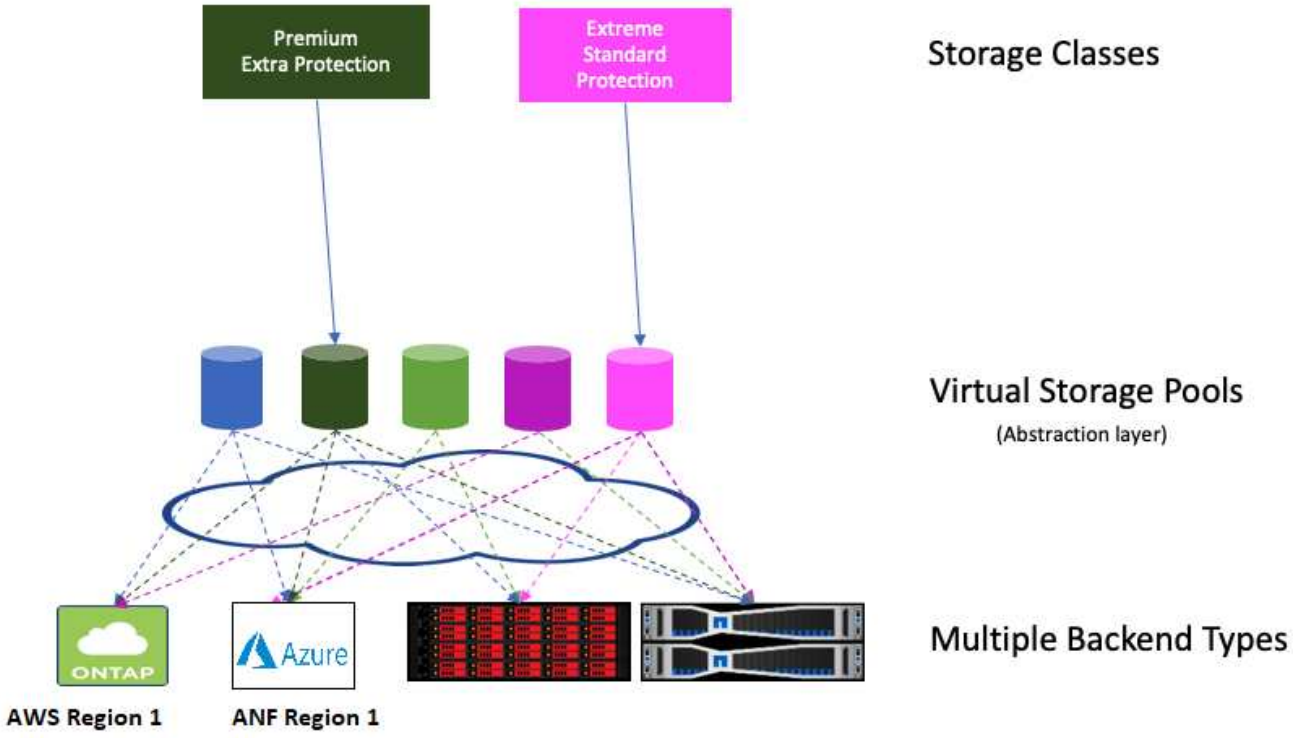
Trident 사용하면 VolumeSnapshots를 사용하여 새로운 PV를 만들 수 있습니다. 이러한 스냅샷에서 PV를 만드는 작업은 지원되는 ONTAP 백엔드에 대한 FlexClone 기술을 사용하여 수행됩니다. 스냅샷에서 PV를 생성할 때 백업 볼륨은 스냅샷의 부모 볼륨의 FlexClone 입니다. 그만큼 `solidfire-san` 드라이버는 Element 소프트웨어 볼륨 복제본을 사용하여 스냅샷에서 PV를 생성합니다. 여기서는 Element 스냅샷에서 복제본을 만듭니다.

Trident 가상 스토리지 풀 구성

가상 풀은 Trident 스토리지 백엔드와 Kubernetes 간에 추상화 계층을 `StorageClasses` 제공합니다. 관리자는 이러한 기능을 사용하여 원하는 기준을 충족하는 데 사용할 물리적 백엔드, 백엔드 풀 또는 백엔드 유형을 지정하지 않고 백엔드에 관계없이 공통의 방식으로 각 백엔드에 대한 위치, 성능 및 보호 등의 측면을 정의할 수 `StorageClass` 있습니다.

가상 풀에 대해 알아보십시오

스토리지 관리자는 Trident 백엔드의 가상 풀을 JSON 또는 YAML 정의 파일로 정의할 수 있습니다.



가상 풀 목록 외부에서 지정된 모든 측면은 백엔드에 대해 전역적이며 모든 가상 풀에 적용되지만, 각 가상 풀은 하나 이상의 측면을 개별적으로 지정할 수 있습니다(백엔드-글로벌 측면 재정의).

- 가상 풀을 정의할 때 백엔드 정의에서 기존 가상 풀의 순서를 재정렬하지 마십시오.
- 기존 가상 풀에 대한 속성을 수정하지 않는 것이 좋습니다. 변경하려면 새 가상 풀을 정의해야 합니다.

대부분의 측면은 백엔드 관련 용어로 지정됩니다. 결정적으로, Aspect 값은 백엔드의 드라이버 외부에 노출되지 않으며 에서 일치시킬 수 없습니다 `StorageClasses`. 대신 관리자는 각 가상 풀에 대해 하나 이상의 레이블을 정의합니다. 각 레이블은 키, 즉 값 쌍이며 레이블은 고유한 백엔드에서 공통일 수 있습니다. 측면과 마찬가지로 레이블을 풀별로 지정하거나 백엔드에 대해 전역으로 지정할 수 있습니다. 미리 정의된 이름과 값이 있는 측면과 달리 관리자는 필요에 따라 레이블 키와 값을 정의할 수 있습니다. 편의를 위해 스토리지 관리자는 가상 풀 및 그룹 볼륨별로 레이블을 레이블별로 정의할 수 있습니다.

가상 풀 레이블은 다음 문자를 사용하여 변경할 수 있습니다.

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

A `StorageClass` 선택기 매개 변수 내의 레이블을 참조하여 사용할 가상 풀을 식별합니다. 가상 풀 선택기는 다음 연산자를 지원합니다.

운영자	예	풀의 레이블 값은 다음과 같아야 합니다.
`='`	성능 = 프리미엄	일치
`!='`	성능!=최고	일치하지 않습니다
"IN"입니다	위치(동쪽, 서쪽)	값 집합에 있어야 합니다
노신입니다	성능 노트(실버, 브론즈)	값 집합에 있지 않습니다
`<key>`	보호	모든 값과 함께 존재합니다
`!<key>`	! 보호	존재하지 않습니다

Trident 볼륨 액세스 그룹 관리

Trident의 사용 방법에 대해 "볼륨 액세스 그룹" 자세히 알아보십시오.

CHAP를 사용하는 경우 이 섹션을 무시하십시오. CHAP는 관리를 단순화하고 아래 설명된 배율 제한을 피하는 것이 좋습니다. 또한 CSI 모드에서 Trident를 사용하는 경우 이 섹션을 무시할 수 있습니다. Trident는 고급 CSI 프로비저닝기로 설치될 때 CHAP를 사용합니다.

볼륨 액세스 그룹에 대해 알아보십시오

Trident는 볼륨 액세스 그룹을 사용하여 프로비저닝하는 볼륨에 대한 액세스를 제어할 수 있습니다. CHAP가 해제된 경우 구성에서 하나 이상의 액세스 그룹 ID를 지정하지 않으면 라는 액세스 그룹을 찾을 것으로 `trident` 예상됩니다.

Trident는 새 볼륨을 구성된 액세스 그룹과 연결하지만 액세스 그룹 자체를 생성하거나 관리하지 않습니다. 액세스 그룹은 스토리지 백엔드를 Trident에 추가하기 전에 존재해야 하며, 해당 백엔드에서 프로비저닝한 볼륨을 마운트할 수 있는 Kubernetes 클러스터의 모든 노드에서 iSCSI IQN을 포함해야 합니다. 대부분의 설치에서는 클러스터의 모든 작업자 노드를 포함합니다.

64개 이상의 노드가 있는 Kubernetes 클러스터의 경우 다중 액세스 그룹을 사용해야 합니다. 각 액세스 그룹에는 최대 64개의 IQN이 포함될 수 있으며 각 볼륨은 4개의 액세스 그룹에 속할 수 있습니다. 최대 4개의 액세스 그룹이 구성되어 있는 경우, 클러스터의 모든 노드에서 최대 256개의 노드 크기로 모든 볼륨에 액세스할 수 있습니다. 볼륨 액세스 그룹에 대한 최신 제한은 을 참조하십시오 "여기".

기본값을 사용하는 구성에서 구성을 수정하는 경우 `trident` 다른 사람을 사용하는 그룹에 대한 액세스 그룹에는 의 ID도 포함됩니다 `trident` 목록에 있는 액세스 그룹.

Trident의 빠른 시작

몇 단계만으로 Trident를 설치하고 스토리지 리소스 관리를 시작할 수 있습니다. 시작하기 전에 를 "Trident 요구사항"검토하십시오.

Docker의 경우 를 "Docker를 위한 Trident"참조하십시오.

1 작업자 노드를 준비합니다

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

"작업자 노드를 준비합니다"

2 **Trident**를 설치합니다

Trident는 다양한 환경 및 조직에 최적화된 다양한 설치 방법 및 모드를 제공합니다.

"Trident를 설치합니다"

3 백엔드를 생성합니다

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

"백엔드를 구성합니다" 확인하십시오

4 **Kubernetes StorageClass**를 생성합니다

Kubernetes StorageClass 객체는 Trident을 프로비저닝자로 지정하고 스토리지 클래스를 생성하여 사용자 지정 가능한 속성으로 볼륨을 프로비저닝할 수 있습니다. Trident은 Trident Provisioner를 지정하는 Kubernetes 오브젝트에 일치하는 스토리지 클래스를 생성합니다.

"스토리지 클래스를 생성합니다"

5 볼륨을 프로비저닝합니다

A_PersistentVolume_(PV)은 Kubernetes 클러스터에서 클러스터 관리자가 프로비저닝한 물리적 스토리지 리소스입니다. PVC(PersistentVolumeClaim_)는 클러스터의 PersistentVolume에 대한 액세스 요청입니다.

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

"볼륨을 프로비저닝합니다"

다음 단계

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

Trident 요구사항

Trident를 설치하기 전에 다음과 같은 일반 시스템 요구 사항을 검토해야 합니다. 특정 백엔드에 추가 요구 사항이 있을 수 있습니다.

Trident에 대한 중요 정보입니다

Trident에 대한 다음 중요 정보를 읽어야 합니다**.**

▼ ** Trident **에 대한 중요 정보입니다

- Trident 26.10은 Kubernetes 1.27부터 1.37까지의 버전을 지원합니다. Kubernetes를 업그레이드하기 전에 아래의 버전별 이미지 목록을 확인하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

지원되는 프런트엔드(오케스트레이터)

Trident는 다음을 비롯한 다수의 컨테이너 엔진 및 오케스트레이터를 지원합니다.

- 베어 메탈 기반 Anthos(VMware) 및 Anthos 1.16
- Kubernetes 1.27 - 1.37
- OpenShift 4.12, 4.14 - 4.22(OpenShift 4.19에서 iSCSI 노드 준비를 사용하려는 경우 지원되는 최소 Trident 버전은 25.06.1입니다.)

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

- Rancher Kubernetes Engine 2(RKE2) v1.28.x - 1.37.x
- NFSv4, iSCSI 및 NVMe/TCP용 Talos OS. 각 프로토콜에는 다음 Talos 시스템 확장이 필요합니다. Sidero Labs에 이러한 요구 사항을 확인하십시오.
  - NFSv4: `siderolabs/nfs-utils`
  - iSCSI: `siderolabs/iscsi-tools`, `siderolabs/multipath-tools` 및 `siderolabs/trident-iscsi-tools`
  - NVMe/TCP: `siderolabs/nvme-cli`, `siderolabs/util-linux-tools` 및

```
siderolabs/trident-iscsi-tools
```

Trident은 Google Kubernetes Engine(GKE), Amazon Elastic Kubernetes Services(EKS), Azure Kubernetes Service(AKS), Mirantis Kubernetes Engine(MKE), VMware Tanzu 포트폴리오를 포함한 기타 완전 관리형 Kubernetes 오퍼링도 지원합니다.

Trident 및 ONTAP를 의 스토리지 공급자로 사용할 수 있습니다"KubeVirt".

> Trident가 설치된 Kubernetes 클러스터를 1.25에서 1.26 이상으로 업그레이드하기 전에 을 참조하십시오."Helm 설치를 업그레이드합니다"

지원되는 백엔드(스토리지)

Trident를 사용하려면 다음과 같은 지원되는 백엔드가 하나 이상 필요합니다.

- NetApp ONTAP용 Amazon FSx
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp 볼륨
- NetApp All SAN 어레이(ASA)
- NetApp 의 전체 또는 제한적 지원을 받는 ONTAP 버전을 실행하는 온프레미스 FAS, AFF 또는 ASA r2(iSCSI, NVMe/TCP 및 FC). 보다 "소프트웨어 버전 지원".
- NetApp HCI/Element 소프트웨어 11 이상

KubeVirt 및 OpenShift 가상화를 위한 Trident 지원

지원되는 스토리지 드라이버:

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

- ONTAP - NAS
- ONTAP-SAN(iSCSI, FCP, NVMe over TCP)
- ONTAP - SAN - 경제성(iSCSI만 해당)

고려 사항:

- OpenShift Virtualization 환경에서 매개 변수(예 `fsType: "ext4"`:)를 갖도록 스토리지 클래스를 업데이트합니다 `fsType`. 필요한 경우 의 매개 변수를 `dataVolumeTemplates` 사용하여 볼륨 모드를 명시적으로 차단하도록 `volumeMode=Block` 설정하여 CDI에 블록 데이터 볼륨을 생성하도록 알립니다.
- 블록 스토리지 드라이버용 *rwx* 액세스 모드:ONTAP-SAN(iSCSI, NVMe/TCP, FC) 및 ONTAP-SAN-Economy(iSCSI) 드라이버는 "volumemode:Block"(원시 장치)에서만 지원됩니다. 이러한 드라이버의 경우 `fstype` 볼륨이 원시 장치 모드로 제공되므로 매개 변수를 사용할 수 없습니다.
- rwx 액세스 모드가 필요한 라이브 마이그레이션 워크플로의 경우 다음 조합이 지원됩니다.
  - NFS+ `volumeMode=Filesystem`
  - iSCSI `volumeMode=Block` + (물리적 장치)
  - NVMe/TCP+ `volumeMode=Block`(원시 장치)

◦ FC+ `volumeMode=Block`(물리적 장치)

Kubernetes 및 OpenShift 호스팅 컨트롤 플레인의 Kubernetes에 대한 Trident 지원

Trident는 Kubernetes on Kubernetes 토폴로지를 완벽하게 지원합니다. 인프라(호스트) 클러스터에서 노드가 KubeVirt 가상 머신으로 실행되는 Kubernetes 클러스터입니다. 이 토폴로지는 OpenShift Virtualization의 OpenShift 호스팅 컨트롤 플레인(HCP) 등 호스팅 컨트롤 플레인 제품의 기반이 됩니다.

ⓘ 이 구성은 이전 버전에서 작동했지만, 완전한 검증은 26.06부터 시작되었습니다.

게스트(테넌트) 클러스터는 KubeVirt CSI 드라이버 (`kubevirt-csi`를 스토리지 공급자로 사용합니다. 게스트 클러스터의 워크로드가 `PersistentVolumeClaim`를 요청하면 KubeVirt CSI 드라이버는 인프라 클러스터에 해당 볼륨을 생성합니다. Trident는 ONTAP에서 볼륨을 프로비저닝하고, 해당 볼륨은 게스트 노드 VM에 핫플러그됩니다. Trident는 인프라 클러스터에서만 실행됩니다. 게스트 클러스터는 KubeVirt CSI 드라이버를 통해 간접적으로 ONTAP 스토리지를 사용합니다.

요구 사항:

- Trident는 지원되는 ONTAP 백엔드를 사용하는 인프라 클러스터에 설치됩니다.
- KubeVirt CSI 드라이버가 게스트 클러스터 볼륨을 매핑할 수 있도록 인프라 클러스터에 Trident 스토리지 클래스가 존재합니다.
- KubeVirt 또는 OpenShift Virtualization이 인프라 클러스터에 배포되고, KubeVirt CSI 드라이버가 각 게스트 클러스터에 설치됩니다.

구성 요소	버전
트라이던트	26.10
Kubernetes(인프라 및 게스트 클러스터)	1.37
KubeVirt	1.9.0
컨테이너화된 데이터 가져오기(CDI)	1.65

고려 사항:

- 게스트 클러스터에서 볼륨은 게스트 노드 VM에서 핫플러그된 블록 장치로 나타납니다.
- KubeVirt CSI 드라이버 워크플로는 게스트 클러스터에 Trident가 필요하지 않습니다. ONTAP 백엔드와 스토리지 클래스는 인프라 클러스터에서 구성합니다.

게스트 클러스터에 **Trident**를 직접 설치합니다.

Trident를 게스트(테넌트) 클러스터에 직접 설치할 수도 있습니다. 스냅샷, 클론, 볼륨 크기 조정 및 가져오기와 같이 `kubevirt-csi`에서 자체적으로 제공하지 않는 전체 Trident 기능 세트가 필요한 경우 게스트 클러스터에 Trident를 설치하십시오.

게스트 클러스터에서는 IP 기반 프로토콜(NFS, iSCSI, NVMe/TCP)을 사용합니다. 게스트 노드는 파이버 채널(FC)에 직접 연결할 수 없는 가상 머신이므로 게스트 클러스터는 FC를 지원하지 않습니다.

피처 요구 사항

아래 표에는 이 Trident 릴리스에서 사용할 수 있는 기능과 이 기능이 지원하는 Kubernetes 버전이 요약되어 있습니다.

피처	**Kubernetes** 버전	기능 게이트가 필요합니까**?**
트라이던트	1.27 - 1.37	아니요
볼륨 스냅샷	1.27 - 1.37	아니요
체적 스냅샷의 PVC	1.27 - 1.37	아니요
iSCSI PV 크기 조정	1.27 - 1.37	아니요
ONTAP 양방향 CHAP	1.27 - 1.37	아니요
동적 내보내기 정책	1.27 - 1.37	아니요
Trident 연산자	1.27 - 1.37	아니요
CSI 토폴로지	1.27 - 1.37	아니요

호스트 운영 체제를 테스트했습니다

Trident는 특정 운영 체제를 공식적으로 지원하지 않지만 다음과 같은 기능이 작동하는 것으로 알려져 있습니다.

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

  NVMe/TCP에는 RHEL 9 이상이 필요합니다.

- AMD64 및 ARM64에서 Ubuntu 22.04 LTS 이상 버전
- Windows Server 2022 를 참조하십시오
- SUSE Linux Enterprise Server (SLES) 15 이상

기본적으로 Trident는 컨테이너에서 실행되므로 모든 Linux 작업자에서 실행됩니다. 그러나 이러한 작업자는 사용 중인 백엔드에 따라 표준 NFS 클라이언트 또는 iSCSI 이니시에이터를 사용하여 Trident에서 제공하는 볼륨을 마운트할 수 있어야 합니다.

트리덴트틀(tridentctl) 유틸리티도 이 리눅스 배포판에서 실행됩니다.

호스트 구성

Kubernetes 클러스터의 모든 작업자 노드는 Pod용으로 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS, iSCSI 또는 NVMe 툴을 설치해야 합니다.

"작업자 노드를 준비합니다"

스토리지 시스템 구성

Trident를 사용하면 스토리지 시스템을 백엔드 구성에서 사용해야 할 수 있습니다.

"백엔드 구성"

Trident 포트

Trident에서는 통신을 위해 특정 포트에 액세스해야 합니다.

"Trident 포트"

컨테이너 이미지 및 해당 Kubernetes 버전

에어갭 설치의 경우 다음 목록은 Trident 설치에 필요한 컨테이너 이미지의 참조입니다. 명령을 사용하여 `tridentctl images` 필요한 컨테이너 이미지 목록을 확인합니다.

아래의 Trident 26.10 이미지 목록에는 Kubernetes 1.37용 이미지가 포함되어 있습니다.

Trident 26.10.1에 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, v1.35.0, v1.36.0, v1.37.0	• docker.io/netapp/trident:26.10.1 • docker.io/netapp/trident-autosupport:26.10 • docker.io/netapp/trident-operator:26.10.1(선택 사항) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.17.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Trident 26.06.2에 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, v1.35.0, v1.36.0	• docker.io/netapp/trident:26.06.2 • docker.io/netapp/trident-autosupport:26.06 • docker.io/netapp/trident-operator:26.06.2(선택 사항) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.18.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Trident를 설치합니다

Trident 설치에 대해 알아봅니다

NetApp는 다양한 환경 및 조직에 Trident를 설치할 수 있도록 다양한 설치 옵션을 제공합니다. Trident 연산자(수동 또는 Helm 사용) 또는 를 사용하여 Trident를 설치할 수 `tridentctl` 있습니다. 이 항목에서는 적합한 설치 프로세스를 선택하는 데 필요한 중요한 정보를 제공합니다.

Trident 26.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다**.**

▼ ** Trident **에 대한 중요 정보입니다

- Kubernetes 1.37는 이제 Trident에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 먼저 업그레이드하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

시작하기 전에

설치 경로에 관계없이 다음 항목이 있어야 합니다.

- 지원되는 버전의 Kubernetes가 실행되고 기능 요구 사항이 활성화된 지원되는 Kubernetes 클러스터에 대한 모든 권한이 필요합니다. 자세한 내용은 "요구 사항"을(를) 참조하십시오.
- 지원되는 NetApp 스토리지 시스템에 대한 액세스
- 모든 Kubernetes 작업자 노드에서 볼륨을 마운트할 수 있습니다.
- 가 설치된 Linux 호스트 `kubectl` (또는 `oc`, OpenShift를 사용하는 경우) 사용하려는 Kubernetes 클러스터를 관리하도록 설치 및 구성한 것입니다.
- 를 클릭합니다 `KUBECONFIG` Kubernetes 클러스터 구성을 가리키도록 설정된 환경 변수입니다.
- Docker Enterprise와 함께 Kubernetes를 사용하는 경우, "다음 단계에 따라 CLI 액세스를 설정합니다".
- 클러스터는 권한이 있는 워크로드를 지원해야 합니다.

에 익숙지 않은 경우 "기본 개념"지금이 바로 익숙해질 때입니다.

설치 방법을 선택합니다

본인에게 맞는 설치 방법을 선택하세요. 결정하기 전에 "방법 간 이동"에 대한 고려 사항도 검토해야 합니다.

Trident 연산자 사용

수동으로 배포하든 Helm을 사용하든, Trident operator는 설치를 간소화하고 Trident 리소스를 동적으로 관리하는 훌륭한 방법입니다. 속성을 사용하여 "Trident 운영자 배포를 사용자 지정합니다"도 할 수 있으며, 이는 `TridentOrchestrator` 커스텀 리소스(CR)에서 가능합니다.

Trident 연산자를 사용하면 다음과 같은 이점이 있습니다.

▼ ** Trident** 개체 생성 ****

Trident 운영자가 Kubernetes 버전에 대해 다음 오브젝트를 자동으로 생성합니다.

- 운영자용 ServiceAccount입니다
- ServiceAccount에 대한 ClusterRole 및 ClusterRoleBinding
- 전용 PodSecurityPolicy(Kubernetes 1.25 이하)
- 작업자 자체

▼ **** 리소스 책임 ****

클러스터 범위의 Trident 운전자가 클러스터 수준에서 Trident 설치와 관련된 리소스를 관리합니다. 이렇게 하면 네임스페이스 범위 연산자를 사용하여 클러스터 범위 리소스를 유지 관리할 때 발생할 수 있는 오류가 줄어듭니다. 이는 자가 복구 및 패치에 필수적입니다.

▼ **** 자동 복구 기능 ****

운영자는 Trident 설치를 모니터링하고 배포가 삭제되거나 실수로 수정된 경우와 같은 문제를 해결하기 위한 조치를 적극적으로 수행합니다. `trident-operator-<generated-id>`CR을 Trident 설치와 연결하는 POD가 `TridentOrchestrator` 생성됩니다. 이렇게 하면 클러스터에 Trident 인스턴스가 하나만 존재하고 해당 설정이 제어되므로 설치가 제대로 이루어지는지 확인할 수 있습니다. 설치 변경(예: 배포 또는 노드 반점 삭제)이 수행되면 운영자가 이를 식별하고 개별적으로 수정합니다.

▼ **** 기존 설치에 대한 간편한 업데이트 ****

기존 배포를 운영자로 쉽게 업데이트할 수 있습니다. 를 편집하기만 하면 됩니다 `TridentOrchestrator` CR을 사용하여 설치를 업데이트합니다.

예를 들어 디버그 로그를 생성하기 위해 Trident를 활성화해야 하는 경우를 생각해 보십시오. 이렇게 하려면 를 패치하여 `TridentOrchestrator` 로 설정합니다 `spec.debug` `true`.

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge
-p '{"spec":{"debug":true}}'
```

이후 `TridentOrchestrator` 이 업데이트되면 운영자가 업데이트를 처리하고 기존 설치를 패치합니다. 이 경우 새 Pod가 생성되어 적절히 설치가 수정될 수 있습니다.

▼ ** Clean ** 재설치

클러스터 범위 Trident 운영자를 사용하면 클러스터 범위 리소스를 깨끗이 제거할 수 있습니다. 사용자는 Trident를 완전히 제거하고 쉽게 다시 설치할 수 있습니다.

▼ ****를 처리하는 **** 자동 **Kubernetes** 업그레이드

클러스터의 Kubernetes 버전을 지원되는 버전으로 업그레이드할 경우 운영자는 기존 Trident 설치를 자동으로 업데이트하고 Kubernetes 버전의 요구사항을 충족하도록 변경합니다.

클러스터가 지원되지 않는 버전으로 업그레이드되면 운영자는 Trident를 설치할 수 없습니다. Trident를 운영자와 함께 이미 설치한 경우 Trident가 지원되지 않는 Kubernetes 버전에 설치되었음을 나타내는 경고가 표시됩니다.

사용 **`tridentctl`**

업그레이드해야 하는 기존 배포가 있거나 배포를 고도로 사용자 지정하려는 경우 고려해야 합니다. 이것이 기존의 Trident 구축 방법입니다.

Trident 리소스에 대한 매니페스트를 생성할 수 있습니다. 여기에는 배포, demonset, 서비스 계정 및 Trident가 설치 과정에서 생성하는 클러스터 역할이 포함됩니다.

22.04 릴리스부터는 Trident를 설치할 때마다 AES 키가 더 이상 재생성되지 않습니다. 이 릴리스에서는 Trident가 설치 전반에 걸쳐 유지되는 새 비밀 개체를 설치합니다. 즉 `tridentctl`, 22.04에서는 이전 버전의 Trident를 제거할 수 있지만 이전 버전에서는 22.04 설치를 제거할 수 없습니다. 적절한 installation_method_를 선택합니다.

설치 모드를 선택합니다

조직에서 요구하는 *installation mode*(Standard, Offline 또는 Remote)를 기반으로 배포 프로세스를 결정합니다.

표준 설치

이 방법은 Trident를 설치하는 가장 쉬운 방법이며 네트워크 제한이 없는 대부분의 환경에서 사용할 수 있습니다. 표준 설치 모드는 기본 레지스트리를 사용하여 필요한 Trident(`docker.io`) 및 CSI(`registry.k8s.io`) 영상을 저장합니다.

표준 모드를 사용하는 경우 Trident 설치 프로그램은 다음을 수행합니다.

- 인터넷을 통해 컨테이너 이미지를 가져옵니다
- 구축 또는 노드 데몬을 생성하여 Kubernetes 클러스터의 모든 적격 노드에서 Trident Pod를 구동합니다

오프라인 설치

오프라인 설치 모드는 공기 박수나 안전한 위치에 필요할 수 있습니다. 이 시나리오에서는 필요한 Trident 및 CSI 이미지를 저장하기 위해 단일 전용 미러된 레지스트리 또는 두 개의 미러링된 레지스트리를 만들 수 있습니다.

레지스트리 구성에 관계없이 CSI 이미지는 하나의 레지스트리에 있어야 합니다.

원격 설치

다음은 원격 설치 프로세스에 대한 상위 수준의 개요입니다.

- Trident를 배포하려는 원격 컴퓨터에 적절한 버전의 를 `kubectl` 배포합니다.
- Kubernetes 클러스터에서 구성 파일을 복사하고 원격 시스템에서 'KUBECONFIG' 환경 변수를 설정합니다.
- 필요한 Kubernetes 클러스터에 연결할 수 있는지 확인하려면 "kubbtl get nodes" 명령을 시작합니다.
- 표준 설치 단계를 사용하여 원격 컴퓨터에서 배포를 완료합니다.

방법 및 모드에 따라 프로세스를 선택합니다

결정을 내린 후 적절한 프로세스를 선택합니다.

방법	설치 모드
Trident 운영자(수동)	"표준 설치" "오프라인 설치"
Trident 운영자(제어)	"표준 설치" "오프라인 설치"
`tridentctl`	"표준 또는 오프라인 설치"

설치 방법 간 이동

설치 방법을 변경할 수 있습니다. 이렇게 하기 전에 다음 사항을 고려하십시오.

- 항상 동일한 방법으로 Trident를 설치 및 제거합니다. 과 함께 를 배포한 경우 `tridentctl` 적절한 버전의 바이너리를 사용하여 Trident를 제거해야 `tridentctl` 합니다. 마찬가지로 운영자와 함께 를 배포하는 경우 CR을 편집하고 `spec.uninstall=true` Trident를 제거하도록 설정해야 `TridentOrchestrator` 합니다.
- 운영자 기반 배포를 제거하고 대신 사용하여 Trident를 배포하려는 경우 `tridentctl` 먼저 을 편집하고 `spec.uninstall=true` Trident를 제거하도록 설정해야 `TridentOrchestrator` 합니다. 그런 다음 `TridentOrchestrator` 및 운영자 배포를 삭제합니다. 그런 다음 을 사용하여 를 설치할 수 `tridentctl` 있습니다.
- 수동 운영자 기반 배포가 있고 Helm 기반 Trident 운영자 배포를 사용하려는 경우 먼저 운영자를 수동으로 제거한 다음 Helm을 설치해야 합니다. 이를 통해 Helm은 필요한 레이블과 주석을 사용하여 Trident 연산자를 배포할 수 있습니다. 이렇게 하지 않으면 Helm 기반 Trident 운영자 배포가 실패하고 레이블 검증 오류와 주석 검증 오류가 발생합니다.
- 만약 당신이 가지고 있다면 `tridentctl` 기반 배포의 경우 Trident 제거하지 않고도 Helm 기반 또는 Operator 기반 배포를 수행할 수 있습니다.

기타 알려진 구성 옵션

VMware Tanzu 포트폴리오 제품에 Trident를 설치하는 경우:

- kubelet-dir 플래그는 kubelet 디렉토리의 위치로 설정해야 합니다. 기본적으로 이 값은 '/var/vcap/data/kubelet'입니다.

  Trident 연산자, Hrom 및 tridentctl 배포에서는 -kubelet -dir 을 사용하여 kubelet 위치를 지정하는 작업이 알려져 있습니다.

Trident 연산자를 사용하여 설치합니다

Trident 연산자 수동 배포(표준 모드)

Trident 운영자를 수동으로 배포하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되지 않은 설치에 적용됩니다. 개인 이미지

레지스트리가 있는 경우 를 "오프라인 배포를 위한 프로세스입니다"사용합니다.

Trident 26.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ ** Trident **에 대한 중요 정보입니다

- Trident 26.10은 Kubernetes 1.27부터 1.37까지 지원합니다. Kubernetes를 업그레이드하기 전에 "Trident 요구사항"에서 버전별 이미지 목록을 확인하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 연산자를 수동으로 구축하고 **Trident**를 설치합니다

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

설치를 시작하기 전에 Linux 호스트에 로그인하여 작동 중인 "지원되는 Kubernetes 클러스터"를 관리하고 있는지, 필요한 권한이 있는지 확인합니다.

OpenShift에서는 뒤에 나오는 모든 예시에서 kubtl 대신 OC를 사용하고, OC login-u system:admin 또는 OC login-u kubbe-admin을 실행하여 먼저 * system:admin * 으로 로그인합니다.

▼ **Details**

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

1단계: **Trident** 설치 프로그램 패키지를 다운로드합니다

Trident 설치 프로그램 패키지에는 Trident 운영자를 배포하고 Trident를 설치하는 데 필요한 모든 것이 포함되어 있습니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 "GitHub의 _Assets_섹션"풉니다.

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2단계: 을 작성합니다 **`TridentOrchestrator` CRD**

생성하다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD). 당신은 만들 것입니다 `TridentOrchestrator` 나중에 사용자 정의 리소스. 적절한 CRD YAML 버전을 사용하세요. `deploy/crds` 생성하기 위해 `TridentOrchestrator` CRD(크레디)

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3단계: **Trident** 연산자를 배포합니다

Trident 설치 프로그램은 운영자를 설치하고 관련 객체를 만드는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 운영자를 배포하고 기본 구성을 사용하여 Trident를 설치하는 쉬운 방법입니다.

- Kubernetes 1.24를 실행하는 클러스터에는 `bundle_pre_1_25.yaml`를 사용합니다.
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

시작하기 전에

- 기본적으로 Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음을 사용하여 생성합니다.

  ```
  kubectl apply -f deploy/namespace.yaml
  ```

- 를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을 사용하여 번들 파일을 생성합니다 `kustomization.yaml`.

  a. 를 생성합니다 `kustomization.yaml` 다음 명령을 사용합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. 다음 명령을 사용하여 번들을 컴파일합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

단계

1. 리소스를 생성하고 연산자를 배포합니다.

```
kubectl create -f deploy/<bundle.yaml>
```

2. 운영자, 배포 및 복제 생성 여부를 확인합니다.

```
kubectl get all -n <operator-namespace>
```

Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Trident 연산자의 여러 배포를 생성하지 마십시오.

4단계: 을 작성합니다 **`TridentOrchestrator` Trident**를 설치합니다

이제 를 만들고 Trident를 설치할 수 `TridentOrchestrator` 있습니다. 선택적으로, 스펙의 속성을 사용할 `TridentOrchestrator` 수 "Trident 설치를 사용자 지정합니다"있습니다.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:         trident
Namespace:
Labels:       <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.10
    Autosupport Proxy:
```

```
    Autosupport Serial Number:
    Debug:                      true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:26.10.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치를 확인합니다

설치를 확인하는 방법에는 여러 가지가 있습니다.

사용 `TridentOrchestrator` 상태

의 상태입니다 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다. 설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운영자가 자체적으로 복구할 수 없습니다. "로그를 확인합니다".

상태	설명
설치 중	운영자가 이 CR을 사용하여 Trident를 `TridentOrchestrator` 설치하는 중입니다.
설치되어 있습니다	Trident가 성공적으로 설치되었습니다.
제거 중	운영자가 Trident를 제거하는 이유는 `spec.uninstall=true`무엇입니까?
제거되었습니다	Trident가 제거되었습니다.
실패했습니다	운영자가 Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 운영자는 이 상태에서 자동으로 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	트리젠터레이터(트리젠터레이터)는 사용하지 않는다. 다른 파일이 이미 있습니다.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                      READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw       6/6     Running   0
1m
trident-node-linux-mr6zc                  2/2     Running   0
1m
trident-node-linux-xrp7w                  2/2     Running   0
1m
trident-node-linux-zh2jt                  2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv         1/1     Running   0
3m
```

사용 **`tridentctl`**

를 사용하여 설치된 Trident 버전을 확인할 수 `tridentctl` 있습니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Trident 연산자 수동 배포(오프라인 모드)

Trident 운영자를 수동으로 배포하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되는 설치에 적용됩니다. 개인 이미지 레지스트리가 없는 경우 를 "표준 배포 프로세스"사용합니다.

Trident에 대한 중요 정보입니다

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ ** Trident **에 대한 중요 정보입니다

- Trident 26.10은 Kubernetes 1.27부터 1.37까지 지원합니다. Kubernetes를 업그레이드하기 전에 "Trident 요구사항"에서 버전별 이미지 목록을 확인하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 연산자를 수동으로 구축하고 Trident를 설치합니다

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

Linux 호스트에 로그인하여 정상적으로 작동하는 "지원되는 Kubernetes 클러스터"를 관리하고 있는지, 그리고 필요한 권한이 있는지 확인하십시오.

OpenShift에서는 뒤에 나오는 모든 예시에서 kubtl 대신 OC를 사용하고, OC login-u system:admin 또는 OC login-u kubbe-admin을 실행하여 먼저 * system:admin * 으로 로그인합니다.

▼ **Details**

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

1단계: **Trident** 설치 프로그램 패키지를 다운로드합니다

Trident 설치 프로그램 패키지에는 Trident 운영자를 배포하고 Trident를 설치하는 데 필요한 모든 것이 포함되어 있습니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 "GitHub의 _Assets_ 섹션"풉니다.

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2단계: 을 작성합니다 **`TridentOrchestrator` CRD**

생성하다 `TridentOrchestrator` 사용자 정의 리소스 정의(CRD). 당신은 만들 것입니다 `TridentOrchestrator` 사용자 정의 리소스는 나중에. 적절한 CRD YAML 버전을 사용하세요. `deploy/crds` 생성하기 위해 `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

3단계: 운영자의 레지스트리 위치를 업데이트합니다

`/deploy/operator.yaml`에서 `image: docker.io/netapp/trident-operator:26.10.1`를 업데이트하여 이미지 레지스트리의 위치를 반영하십시오. xref:{relative_path}../trident-get-started/requirements.html#container-images-and-corresponding-kubernetes-versions["Trident 및 CSI 이미지"]는 하나의 레지스트리 또는 여러 레지스트리에 위치할 수 있지만, 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예를 들면 다음과 같습니다.

- `image: <your-registry>/trident-operator:26.10.1` 이미지가 모두 하나의 레지스트리에 있는 경우.
- `image: <your-registry>/netapp/trident-operator:26.10.1` Trident 이미지가 CSI 이미지와 다른 레지스트리에 있는 경우.

4단계: **Trident** 연산자를 배포합니다

Trident 설치 프로그램은 운영자를 설치하고 관련 객체를 만드는 데 사용할 수 있는 번들 파일을 제공합니다. 번들 파일은 운영자를 배포하고 기본 구성을 사용하여 Trident를 설치하는 쉬운 방법입니다.

- Kubernetes 1.24를 실행하는 클러스터에는 `bundle_pre_1_25.yaml`를 사용합니다.
- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

시작하기 전에

- 기본적으로 Trident 설치 관리자는 에 연산자를 배포합니다 `trident` 네임스페이스. 를 누릅니다 `trident` 네임스페이스가 없습니다. 다음을 사용하여 생성합니다.

  ```
  kubectl apply -f deploy/namespace.yaml
  ```

- 를 제외한 네임스페이스에 연산자를 배포합니다 `trident` 네임스페이스, 업데이트 `serviceaccount.yaml`, `clusterrolebinding.yaml` 및 `operator.yaml` 을 사용하여 번들 파일을 생성합니다 `kustomization.yaml`.

  a. 를 생성합니다 `kustomization.yaml` 다음 명령을 사용합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

b. 다음 명령을 사용하여 번들을 컴파일합니다. 여기서 _<bundle.yaml>_is `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전을 기반으로 합니다.

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

단계

1. 리소스를 생성하고 연산자를 배포합니다.

```
kubectl create -f deploy/<bundle.yaml>
```

2. 운영자, 배포 및 복제 생성 여부를 확인합니다.

```
kubectl get all -n <operator-namespace>
```

Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Trident 연산자의 여러 배포를 생성하지 마십시오.

5단계: 에서 이미지 레지스트리 위치를 업데이트합니다 **`TridentOrchestrator`**

귀사의 "Trident 및 CSI 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 업데이트 `deploy/crds/tridentorchestrator_cr.yaml` 레지스트리 구성에 따라 추가 위치 사양을 추가하려면 다음을 수행합니다.

하나의 레지스트리에 있는 이미지

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

다른 레지스트리의 이미지

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

6단계: 을 작성합니다 **`TridentOrchestrator`** **Trident**를 설치합니다

이제 를 만들고 Trident를 설치할 수 `TridentOrchestrator` 있습니다. 선택적으로, 스펙의 속성을 추가로 사용할 `TridentOrchestrator` 수 "Trident 설치를 사용자 지정합니다"있습니다. 다음 예에서는 Trident 및 CSI 이미지가 다른 레지스트리에 있는 설치를 보여 줍니다.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.10.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.10.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

설치를 확인합니다

설치를 확인하는 방법에는 여러 가지가 있습니다.

사용 `TridentOrchestrator` 상태

의 상태입니다 `TridentOrchestrator` 설치가 성공적으로 완료되었는지 여부를 나타내고 설치된 Trident의 버전을 표시합니다. 설치하는 동안 의 상태입니다 `TridentOrchestrator` 변경 시작 `Installing` 를 선택합니다 `Installed`. 을(를) 관찰하면 `Failed` 상태 및 운영자가 자체적으로 복구할 수 없습니다. "로그를 확인합니다".

상태	설명
설치 중	운영자가 이 CR을 사용하여 Trident를 `TridentOrchestrator` 설치하는 중입니다.
설치되어 있습니다	Trident가 성공적으로 설치되었습니다.
제거 중	운영자가 Trident를 제거하는 이유는 `spec.uninstall=true`무엇입니까?
제거되었습니다	Trident가 제거되었습니다.
실패했습니다	운영자가 Trident를 설치, 패치, 업데이트 또는 제거할 수 없습니다. 운영자는 이 상태에서 자동으로 복구를 시도합니다. 이 상태가 지속되면 문제 해결이 필요합니다.
업데이트 중	운영자가 기존 설치를 업데이트하고 있습니다.
오류	트리젠터레이터(트리젠터레이터)는 사용하지 않는다. 다른 파일이 이미 있습니다.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw   6/6     Running   0
1m
trident-node-linux-mr6zc              2/2     Running   0
1m
trident-node-linux-xrp7w              2/2     Running   0
1m
trident-node-linux-zh2jt              2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0
3m
```

사용 **tridentctl**

를 사용하여 설치된 Trident 버전을 확인할 수 `tridentctl` 있습니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

H제어(표준 모드)를 사용하여 **Trident** 연산자 배포

Trident 연산자를 배포하고 Helm을 사용하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되지 않은 설치에 적용됩니다. 개인 이미지 레지스트리가 있는 경우 를 "오프라인 배포를 위한 프로세스입니다"사용합니다.

Trident 26.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다**.**

▼ ** Trident **에 대한 중요 정보입니다

- Trident 26.10은 Kubernetes 1.27부터 1.37까지 지원합니다. Kubernetes를 업그레이드하기 전에 "Trident 요구사항"에서 버전별 이미지 목록을 확인하십시오.
- Tridentis multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 연산자를 배포하고 **Helm**을 사용하여 **Trident**를 설치합니다

Trident 사용 "Helm 차트" Trident 연산자를 구축하고 Trident를 한 번에 설치할 수 있습니다.

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

"구축 사전 요구 사항" 외에도 "Helm 버전 3"이(가) 필요합니다.

단계

1. Trident Helm 저장소 추가:

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

2. `helm install`를 사용하고 다음 예시와 같이 배포 이름을 지정하십시오. 여기서 `100.2610.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace <trident-namespace>
```

Trident에 대한 네임스페이스를 이미 만든 경우에는 '--create-namespace' 매개 변수가 추가 네임스페이스를 만들지 않습니다.

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션을 선택합니다	설명
`--values` (또는 `-f`)	재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.
`--set`	명령줄에 overrides를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2610.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

구성 옵션

이 표와 `values.yaml` 제어 차트의 일부인 파일 에는 키 목록과 해당 기본값이 나와 있습니다.

옵션을 선택합니다	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블입니다	
`podAnnotations`	창 주석	
`deploymentAnnotations`	배포 주석	
`tolerations`	POD 지정에 대한 공차	

옵션을 선택합니다	설명	기본값

`affinity`	POD 할당에 대한 선호도	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		values.yaml 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장합니다.

`tridentControllerPluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 컨트롤러 Pod 및 노드 포드 이해 를 참조하십시오.	

옵션을 선택합니다	설명	기본값
'imageRegistry' 입니다	, `trident` 및 기타 이미지의 레지스트리를 `trident-operator` 식별합니다. 기본값을 그대로 사용하려면 비워 두십시오. 중요: 전용 리포지토리에 Trident를 설치할 때 스위치를 사용하여 리포지토리 위치를 지정하는 경우 `imageRegistry` 리포지토리 경로에서 를 사용하지 마십시오 `/netapp/`.	""
`imagePullPolicy`	에 대한 이미지 풀 정책을 설정합니다 `trident-operator`.	`IfNotPresent`
'imagePullSecrets'	의 이미지 풀 비밀을 설정합니다 `trident-operator`, `trident`및 기타 이미지.	
쿠벨레트디렉토리	kubelet 내부 상태의 호스트 위치를 재정의할 수 있습니다.	`"/var/lib/kubelet"`
`operatorLogLevel`	Trident 연산자의 로그 수준을 다음으로 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`
`operatorDebug`	Trident 연산자의 로그 수준을 디버깅으로 설정할 수 있습니다.	"참"입니다
`operatorImage`	에 대한 이미지를 완전히 재정의할 수 있습니다 `trident-operator`.	""
`operatorImageTag`	의 태그를 재정의할 수 있습니다 `trident-operator` 이미지.	""
`tridentIPv6`	Trident가 IPv6 클러스터에서 작동하도록 설정합니다.	거짓입니다
`tridentK8sTimeout`	대부분의 Kubernetes API 작업에 대한 기본 30초 시간 초과(0이 아닌 경우 초)를 재정의합니다.	`0`
`tridentHttpRequestTimeout`	에서는 HTTP 요청에 대한 기본 90초 제한 시간을 재정의합니다 `0s` 제한 시간 동안 무한 지속 시간입니다. 음수 값은 허용되지 않습니다.	`"90s"`
`tridentSilenceAutosupport`	Trident 정기 AutoSupport 보고를 비활성화할 수 있습니다.	거짓입니다
`tridentExcludeAutosupport`	`true`로 설정된 경우, Trident는 설치 중에 AutoSupport 컨테이너를 설정하지 않습니다.	거짓입니다

옵션을 선택합니다	설명	기본값
tridentAutosupportImageTag	Trident AutoSupport 컨테이너에 대한 이미지 태그를 재정의할 수 있습니다.	<version>
tridentAutosupportProxy	HTTP 프록시를 통해 Trident AutoSupport 컨테이너를 phone home으로 설정합니다.	""
tridentLogFormat	Trident 로깅 형식 (text 또는 json)을 설정합니다.	"text"
tridentDisableAuditLog	Trident 감사 로거를 비활성화합니다.	"참"입니다
tridentLogLevel	Trident의 로그 수준을 , , debug, info, warn error 또는 fatal 로 설정할 수 trace 있습니다.	"info"
tridentDebug	Trident 의 로그 수준을 설정할 수 있습니다. debug . 노드 상태 점검(NHC) 연산자와 통합하여 강제 분리 프로세스를 자동화할 수 있습니다. 자세한 내용은 다음을 참조하세요."Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화" .	거짓입니다
tridentLogWorkflows	추적 로깅 또는 로그 억제를 위해 특정 Trident 워크플로우를 사용할 수 있습니다.	""
tridentLogLayers	추적 로깅 또는 로그 억제를 위해 특정 Trident 계층을 사용할 수 있습니다.	""
트리덴티이미지	Trident에 대한 이미지의 전체 덮어쓰기를 허용합니다.	""
tridentImageTag	Trident에 대한 이미지 태그를 재정의할 수 있습니다.	""
tridentProbePort	Kubernetes 활성/준비 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""
windows	Windows 작업자 노드에 Trident를 설치할 수 있습니다.	거짓입니다
enableForceDetach	힘 분리 기능을 활성화합니다.	거짓입니다

옵션을 선택합니다	설명	기본값
`enableDataLIFRefresh`	LIF가 추가, 제거되거나 새 주소가 할당될 때 게시된 볼륨의 NVMe 데이터 LIF를 새로 고칩니다. 통합 ONTAP의 `ontap-san` 및 `sanType: nvme`와 ASA r2의 NVMe/TCP에 적용됩니다. 자세한 내용은 "NVMe 데이터 LIF 새로 고침"을 참조하십시오.	거짓입니다
`excludePodSecurityPolicy`	운영자 POD 보안 정책을 생성할 수 없습니다.	거짓입니다
`cloudProvider`	AKS 클러스터에서 관리 ID 또는 클라우드 ID를 사용할 때는 "`Azure`"(으)로, EKS 클러스터에서 클라우드 ID를 사용할 때는 "`AWS`"(으)로, GKE에서 클라우드 ID를 사용하거나 OpenShift에서 Workload Identity Federation을 사용할 때는 "`GCP`"(으)로 설정합니다.	""
`cloudIdentity`	AKS 클러스터에서 클라우드 ID를 사용하는 경우 워크로드 ID("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx")로 설정하십시오. EKS 클러스터에서 클라우드 ID를 사용하는 경우 AWS IAM 역할("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'")로 설정하십시오. GKE에서 클라우드 ID를 사용하는 경우 GKE 클라우드 ID 주석으로 설정하십시오. OpenShift 워크로드 ID 페더레이션의 경우에는 설정하지 마십시오.	""
`gcpWIFAudience`	OpenShift의 Google Cloud Workload Identity Federation과 함께 사용되는 프로젝션된 서비스 계정 토큰의 대상입니다. 공급자의 정규 리소스 이름을 사용하십시오. 자세한 내용은 "워크로드 ID 페더레이션을 사용하여 Google Cloud NetApp Volumes 구성"을(를) 참조하십시오.	""
`iscsiSelfHealingInterval`	iSCSI 자동 복구가 호출되는 간격입니다.	`5m0s`
`iscsiSelfHealingWaitTime`	iSCSI 자체 복구가 로그아웃과 후속 로그인을 수행하여 부실 세션을 해결하려는 시도를 시작한 이후의 기간입니다.	`7m0s`

옵션을 선택합니다	설명	기본값
`nodePrep`	Trident 지정된 데이터 저장 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 합니다. 현재, **`iscsi`** 지원되는 유일한 값입니다. 참고: OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	
`enableConcurrency`	Trident 컨트롤러의 동시 작동을 지원하여 처리량을 향상시킵니다. 지원되는 드라이버에 대한 자세한 내용은 링크: 컨트롤러 확장성 문서를 참조하십시오.	거짓
`k8sAPIQPS`	Kubernetes API 서버와 통신하는 동안 컨트롤러가 사용하는 초당 쿼리 수(QPS) 제한입니다. 버스트 값은 QPS 값에 따라 자동으로 설정됩니다.	`100`; 선택 사항

옵션을 선택합니다	설명	기본값

`resources`	Trident 컨트롤러, 노드 및 운영자 포드에 대한 Kubernetes 리소스 제한 및 요청을 설정합니다. Kubernetes에서 리소스 할당을 관리하기 위해 각 컨테이너와 사이드카에 대한 CPU와 메모리를 구성할 수 있습니다.
	
	리소스 요청 및 제한 구성에 대한 자세한 내용은 다음을 참조하세요. "포드 및 컨테이너에 대한 리소스 관리" .
	
	• 컨테이너나 필드의 이름을 변경하지 마세요.
	• 들여쓰기를 변경하지 마세요. YAML 들여쓰기는 올바른 구문 분석을 위해 중요합니다.
	
	• 기본적으로 제한은 적용되지 않습니다. 요청에만 기본값이 있으며, 지정하지 않으면 자동으로 적용됩니다.
	• 컨테이너 이름은 포드 사양에 나타나는 대로 나열됩니다.
	• 사이드카는 각 주요 컨테이너 아래에 나열되어 있습니다.
	• TORC를 확인하세요 `status.CurrentInstallationParams` 현재 적용된 값을 보려면 필드를 클릭하세요.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`

옵션을 선택합니다	설명	기본값
`httpsMetrics`	Prometheus 메트릭 엔드포인트에 HTTPS를 활성화합니다.	거짓
호스트 네트워크	Trident 컨트롤러에 대한 호스트 네트워킹을 활성화합니다. 이 기능은 멀티 홈 네트워크에서 프런트엔드와 백엔드 트래픽을 분리할 때 유용합니다.	거짓

컨트롤러 **Pod** 및 노드 포드 이해

Trident는 단일 컨트롤러 Pod와 클러스터의 각 작업자 노드에서 노드 Pod로 실행됩니다. 노드 포드는 Trident 볼륨을 마운트하려는 호스트에서 실행되고 있어야 합니다.

쿠버네티스 "노드 선택기" 및 "관용과 오해" 포드를 특정 노드 또는 기본 노드에서 실행하도록 제한하는 데 사용됩니다. ControllerPlugin과 을 사용합니다 `NodePlugin`구속 조건과 덮어쓰기를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

H제어(오프라인 모드)를 사용하여 **Trident** 연산자 배포

Trident 연산자를 배포하고 Helm을 사용하여 Trident를 설치할 수 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되는 설치에 적용됩니다. 개인 이미지 레지스트리가 없는 경우 를 "표준 배포 프로세스"사용합니다.

Trident 26.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다**.**

▼ ** Trident **에 대한 중요 정보입니다

- Trident 26.10은 Kubernetes 1.27부터 1.37까지 지원합니다. Kubernetes를 업그레이드하기 전에 "Trident 요구사항"에서 버전별 이미지 목록을 확인하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

Trident 연산자를 배포하고 **Helm**을 사용하여 **Trident**를 설치합니다

Trident 사용 "Helm 차트" Trident 연산자를 구축하고 Trident를 한 번에 설치할 수 있습니다.

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

"구축 사전 요구 사항" 외에도 "Helm 버전 3"이(가) 필요합니다.

> 전용 리포지토리에 Trident를 설치할 때 스위치를 사용하여 리포지토리 위치를 지정하는 경우 `imageRegistry` 리포지토리 경로에서 를 사용하지 마십시오 `/netapp/`.

단계

1. Trident Helm 저장소 추가:

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

2. `helm install`를 사용하여 배포 및 이미지 레지스트리 위치에 이름을 지정하십시오. "Trident 및 CSI 이미지"는 하나의 레지스트리 또는 여러 레지스트리에 위치할 수 있지만, 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다. 예시에서 `100.2610.0`는 설치하려는 Trident 버전입니다.

하나의 레지스트리에 있는 이미지

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

다른 레지스트리의 이미지

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.10.1 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.10
--set tridentImage=<your-registry>/trident:26.10.1 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

> Trident에 대한 네임스페이스를 이미 만든 경우에는 '--create-namespace' 매개 변수가 추가 네임스페이스를 만들지 않습니다.

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

설치 중에 구성 데이터를 전달합니다

설치 중에 구성 데이터를 전달하는 방법에는 두 가지가 있습니다.

옵션을 선택합니다	설명
`--values` (또는 `-f`)	재정의가 있는 YAML 파일을 지정합니다. 이 옵션은 여러 번 지정할 수 있으며 가장 오른쪽 파일이 우선 적용됩니다.

옵션을 선택합니다	설명
`--set`	명령줄에 overrides를 지정합니다.

예를 들어, `debug`의 기본값을 변경하려면 다음 명령을 실행하십시오. 여기서 `100.2610.0`는 설치하려는 Trident 버전입니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

nodePrep 값을 추가하려면 다음 명령을 실행합니다.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

구성 옵션

이 표와 `values.yaml` 제어 차트의 일부인 파일 에는 키 목록과 해당 기본값이 나와 있습니다.



values.yaml 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장합니다.

옵션을 선택합니다	설명	기본값
`nodeSelector`	POD 할당을 위한 노드 레이블입니다	
`podAnnotations`	창 주석	
`deploymentAnnotations`	배포 주석	
`tolerations`	POD 지정에 대한 공차	

옵션을 선택합니다	설명	기본값

`affinity`	POD 할당에 대한 선호도	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		values.yaml 파일에서 기본 선호도를 제거하지 마십시오. 사용자 지정 선호도를 제공하려면 기본 선호도를 확장합니다.

`tridentControllerPluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 "컨트롤러 Pod 및 노드 포드 이해" 를 참조하십시오.	
`tridentControllerPluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 "컨트롤러 Pod 및 노드 포드 이해" 를 참조하십시오.	
`tridentNodePluginNodeSelector`	Pod용 추가 노드 선택기 을 참조하십시오 "컨트롤러 Pod 및 노드 포드 이해" 를 참조하십시오.	

옵션을 선택합니다	설명	기본값
`tridentNodePluginTolerations`	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 을 참조하십시오 "컨트롤러 Pod 및 노드 포드 이해" 를 참조하십시오.	
'imageRegistry'입니다	, `trident` 및 기타 이미지의 레지스트리를 `trident-operator` 식별합니다. 기본값을 그대로 사용하려면 비워 두십시오. 중요: 전용 리포지토리에 Trident를 설치할 때 스위치를 사용하여 리포지토리 위치를 지정하는 경우 `imageRegistry` 리포지토리 경로에서 를 사용하지 마십시오 `/netapp/`.	""
`imagePullPolicy`	에 대한 이미지 풀 정책을 설정합니다 `trident-operator`.	`IfNotPresent`
'imagePullSecrets'	의 이미지 풀 비밀을 설정합니다 `trident-operator`, `trident`및 기타 이미지.	
쿠벨레트디렉토리	kubelet 내부 상태의 호스트 위치를 재정의할 수 있습니다.	`"/var/lib/kubelet"`
`operatorLogLevel`	Trident 연산자의 로그 수준을 다음으로 설정할 수 있습니다. `trace`, `debug`, `info`, `warn`, `error`, 또는 `fatal`.	`"info"`
`operatorDebug`	Trident 연산자의 로그 수준을 디버깅으로 설정할 수 있습니다.	"참"입니다
`operatorImage`	에 대한 이미지를 완전히 재정의할 수 있습니다 `trident-operator`.	""
`operatorImageTag`	의 태그를 재정의할 수 있습니다 `trident-operator` 이미지.	""
`tridentIPv6`	Trident가 IPv6 클러스터에서 작동하도록 설정합니다.	거짓입니다
`tridentK8sTimeout`	대부분의 Kubernetes API 작업에 대한 기본 180초 시간 제한을 재정의합니다(0이 아닌 경우 초 단위). 그만큼 `tridentK8sTimeout` 매개변수는 Trident 설치에만 적용됩니다.	`180`

옵션을 선택합니다	설명	기본값
`tridentHttpRequestTimeout`	에서는 HTTP 요청에 대한 기본 90초 제한 시간을 재정의합니다 `0s` 제한 시간 동안 무한 지속 시간입니다. 음수 값은 허용되지 않습니다.	`"90s"`
`tridentSilenceAutosupport`	Trident 정기 AutoSupport 보고를 비활성화할 수 있습니다.	거짓입니다
`tridentExcludeAutosupport`	`true`로 설정된 경우, Trident는 설치 중에 AutoSupport 컨테이너를 설정하지 않습니다.	거짓입니다
`tridentAutosupportImageTag`	Trident AutoSupport 컨테이너에 대한 이미지 태그를 재정의할 수 있습니다.	`<version>`
`tridentAutosupportProxy`	HTTP 프록시를 통해 Trident AutoSupport 컨테이너를 phone home으로 설정합니다.	""
`tridentLogFormat`	Trident 로깅 형식 (`text` 또는 `json`)을 설정합니다.	`"text"`
`tridentDisableAuditLog`	Trident 감사 로거를 비활성화합니다.	"참"입니다
`tridentLogLevel`	Trident의 로그 수준을 , , `debug`, `info`, `warn` `error` 또는 `fatal` 로 설정할 수 `trace` 있습니다.	`"info"`
`tridentDebug`	Trident의 로그 수준을 로 설정할 수 `debug` 있습니다.	거짓입니다
`tridentLogWorkflows`	추적 로깅 또는 로그 억제를 위해 특정 Trident 워크플로우를 사용할 수 있습니다.	""
`tridentLogLayers`	추적 로깅 또는 로그 억제를 위해 특정 Trident 계층을 사용할 수 있습니다.	""
트리덴티이미지	Trident에 대한 이미지의 전체 덮어쓰기를 허용합니다.	""
`tridentImageTag`	Trident에 대한 이미지 태그를 재정의할 수 있습니다.	""
`tridentProbePort`	Kubernetes 활성/준비 프로브에 사용되는 기본 포트를 재정의할 수 있습니다.	""
`windows`	Windows 작업자 노드에 Trident를 설치할 수 있습니다.	거짓입니다

옵션을 선택합니다	설명	기본값
`enableForceDetach`	강제 분리 기능을 활성화할 수 있습니다. 노드 상태 점검(NHC) 연산자와 통합하여 강제 분리 프로세스를 자동화할 수 있습니다. 자세한 내용은 다음을 참조하세요 ."Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화" .	거짓입니다
`enableDataLIFRefresh`	LIF가 추가, 제거되거나 새 주소가 할당될 때 게시된 볼륨의 NVMe 데이터 LIF를 새로 고칩니다. 통합 ONTAP의 `ontap-san` 및 `sanType: nvme`와 ASA r2의 NVMe/TCP에 적용됩니다. 자세한 내용은 "NVMe 데이터 LIF 새로 고침"을 참조하십시오.	거짓입니다
`excludePodSecurityPolicy`	운영자 POD 보안 정책을 생성할 수 없습니다.	거짓입니다
`nodePrep`	Trident가 Kubernetes 클러스터의 노드를 준비하여 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리할 수 있도록 합니다. * 현재 `iscsi` 지원되는 유일한 값입니다. * OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	

옵션을 선택합니다	설명	기본값

`resources`	Trident 컨트롤러, 노드 및 운영자 포드에 대한 Kubernetes 리소스 제한 및 요청을 설정합니다. Kubernetes에서 리소스 할당을 관리하기 위해 각 컨테이너와 사이드카에 대한 CPU와 메모리를 구성할 수 있습니다.
	
	리소스 요청 및 제한 구성에 대한 자세한 내용은 다음을 참조하세요. "포드 및 컨테이너에 대한 리소스 관리" .
	
	• 컨테이너나 필드의 이름을 변경하지 마세요.
	• 들여쓰기를 변경하지 마세요. YAML 들여쓰기는 올바른 구문 분석을 위해 중요합니다.
	
	• 기본적으로 제한은 적용되지 않습니다. 요청에만 기본값이 적용됩니다.
	• 컨테이너 이름은 포드 사양에 나타나는 대로 나열됩니다.
	• 사이드카는 각 주요 컨테이너 아래에 나열되어 있습니다.
	• TORC를 확인하세요 `status.CurrentInstallationParams` 현재 적용된 값을 보려면 필드를 클릭하세요.
		<pre>resources:
		  controller:
		    trident-main:
		      requests:
		        cpu: 10m
		        memory: 80Mi
		      limits:
		          cpu:
		          memory:
		    csi-provisioner:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-attacher:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-resizer:
		      requests:
		        cpu: 3m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-snapshotter:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    trident-
		autosupport:
		      requests:
		        cpu: 1m
		        memory: 30Mi
		      limits:
		        cpu:
		        memory:</pre>

Trident 운영자 설치를 사용자 지정합니다

Trident 연산자를 사용하면 사양의 속성을 사용하여 Trident 설치를 사용자 정의할 수 `TridentOrchestrator` 있습니다. 허용되는 인수를 넘어 설치를 사용자 지정하려면 `TridentOrchestrator` 을 사용하여 `tridentctl` 사용자 지정 YAML 매니페스트를 생성하여 필요에 따라 수정할 수 있습니다.

컨트롤러 Pod 및 노드 포드 이해

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

쿠버네티스 "노드 선택기" 및 "관용과 오해" 포드를 특정 노드 또는 기본 노드에서 실행하도록 제한하는 데 사용됩니다. ControllerPlugin과 을 사용합니다 `NodePlugin`구속 조건과 덮어쓰기를 지정할 수 있습니다.

- 컨트롤러 플러그인은 스냅샷 및 크기 조정과 같은 볼륨 프로비저닝 및 관리를 처리합니다.
- 노드 플러그인은 스토리지에 노드를 연결하는 작업을 처리합니다.

구성 옵션

> `spec.namespace` Trident가 설치된 네임스페이스를 나타내기 위해 에 `TridentOrchestrator` 지정됩니다. 이 매개 변수 * 는 Trident 설치 후 업데이트할 수 없습니다 *. 이렇게 하면 `TridentOrchestrator` 상태가 로 `Failed`변경됩니다. Trident는 네임스페이스 간에 마이그레이션되지 않습니다.

이 표에 자세히 나와 있습니다 `TridentOrchestrator` 속성.

매개 변수	설명	기본값
`namespace`	Trident를 설치할 네임스페이스입니다	`"default"`
"버그"	Trident에 대해 디버깅을 활성화합니다	거짓입니다
`enableForceDetach`	`ontap-san`, `ontap-san-economy` , `ontap-nas` , 그리고 `ontap-nas-economy` 오직. Kubernetes Non-Graceful Node Shutdown(NGNS)과 함께 작동하여 클러스터 관리자가 노드가 비정상적으로 될 경우 마운트된 볼륨이 있는 작업 부하를 새 노드로 안전하게 마이그레이션할 수 있는 기능을 제공합니다. 자세한 내용은 다음을 참조하세요."Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화" .	거짓입니다
`enableDataLIFRefresh`	LIF가 추가, 제거되거나 새 주소가 할당될 때 게시된 볼륨에 대한 NVMe 데이터 LIF를 새로 고칠 수 있습니다. 통합 ONTAP의 `ontap-san`과 `sanType: nvme` 및 ASA r2의 NVMe/TCP에 적용됩니다. 자세한 내용은 "NVMe 데이터 LIF 새로 고침"을 참조하십시오.	거짓입니다
`windows`	를 로 설정합니다 `true` Windows 작업자 노드에 설치할 수 있습니다.	거짓입니다

매개 변수	설명	기본값
`cloudProvider`	AKS 클러스터에서 관리되는 ID 또는 클라우드 ID를 사용할 때 로 `"Azure"` 설정합니다. EKS 클러스터에서 클라우드 ID를 사용할 때 로 `"AWS"` 설정합니다. GKE 클러스터에서 클라우드 ID를 사용할 때 로 `"GCP"` 설정합니다.	`""`
`cloudIdentity`	AKS 클러스터에서 클라우드 ID를 사용할 때 워크로드 ID("Azure.workload.identity/client-id: xxxxxxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx")로 설정합니다. EKS 클러스터에서 클라우드 ID를 사용할 때 AWS IAM 역할("'eks.amazonaws.com/role-arn: arn:AWS:IAM::123456:role/Trident-role'")로 설정합니다. GKE 클러스터에서 클라우드 ID를 사용하는 경우 클라우드 ID("'iam.gke.io/gc-service-account:xxxx@mygcpproject.iam.gserviceaccount.com') 로 설정합니다.	`""`
`gcpWIFAudience`	OpenShift의 Google Cloud Workload Identity Federation과 함께 사용되는 프로젝션된 서비스 계정 토큰의 대상입니다. 공급자의 정규 리소스 이름을 사용하십시오. 자세한 내용은 "워크로드 ID 페더레이션을 사용하여 Google Cloud NetApp Volumes 구성"을(를) 참조하십시오.	`""`
`IPv6`	IPv6를 통해 Trident를 설치합니다	거짓
`k8sTimeout`	Kubernetes 작업에 대한 시간 초과입니다. 그만큼 `k8sTimeout` 매개변수는 Trident 설치에만 적용됩니다.	`180sec`
`silenceAutosupport`	AutoSupport 번들을 NetApp로 보내지 마십시오 자동으로 계층화	거짓입니다
자동 지원 이미지	AutoSupport 텔레메트리 컨테이너 이미지입니다	`"netapp/trident-autosupport:26.10"`
자동 지원 프록시	AutoSupport를 보내는 프록시의 주소/포트입니다 원격 측정	`"http://proxy.example.com:8888"`
"제거"를 선택합니다	Trident를 제거하는 데 사용되는 플래그입니다	거짓입니다
`logFormat`	사용할 Trident 로깅 형식 [text, json]	`"text"`
트리덴티이미지	설치할 Trident 이미지	`"netapp/trident:26.10.1"`
'imageRegistry'입니다	형식의 내부 레지스트리 경로입니다 `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
쿠벨레트디렉토리	호스트의 kubelet 디렉토리에 대한 경로입니다	`"/var/lib/kubelet"`
`wipeout`	Trident의 전체 제거를 수행하기 위해 삭제할 리소스 목록입니다	
'imagePullSecrets'	내부 레지스트리에서 이미지를 가져올 수 있는 비밀	

매개 변수	설명	기본값
`imagePullPolicy`	Trident 운영자의 이미지 풀 정책을 설정합니다. 유효한 값은 다음과 같습니다. `Always` 항상 이미지를 당깁니다. `IfNotPresent` 이미지가 아직 노드에 없는 경우에만 이미지를 가져옵니다. `Never` 이미지를 당기지 않습니다.	`IfNotPresent`
'controllerPluginNodeSelector'	Pod용 추가 노드 선택기 는 와 동일한 형식을 따릅니다 `pod.spec.nodeSelector`.	기본값 없음, 선택 사항
'컨트롤러 PluginTolerations'	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 는 와 동일한 형식을 따릅니다 `pod.spec.Tolerations`.	기본값 없음, 선택 사항
노드플러거노드 선택기	Pod용 추가 노드 선택기 는 와 동일한 형식을 따릅니다 `pod.spec.nodeSelector`.	기본값 없음, 선택 사항
노드 PluginTolerations	Pod에 대한 Kubernetes 허용 설정을 재정의합니다. 는 와 동일한 형식을 따릅니다 `pod.spec.Tolerations`.	기본값 없음, 선택 사항
`nodePrep`	Trident가 Kubernetes 클러스터의 노드를 준비하여 지정된 데이터 스토리지 프로토콜을 사용하여 볼륨을 관리할 수 있도록 합니다. * 현재 `iscsi` 지원되는 유일한 값입니다. * OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.	
`k8sAPIQPS`	Kubernetes API 서버와 통신하는 동안 컨트롤러가 사용하는 초당 쿼리 수(QPS) 제한입니다. 버스트 값은 QPS 값에 따라 자동으로 설정됩니다.	`100`; 선택 사항
`enableConcurrency`	Trident 컨트롤러의 동시 작동을 지원하여 처리량을 향상시킵니다. 지원되는 드라이버에 대한 자세한 내용은 링크: 컨트롤러 확장성 문서를 참조하십시오.	`enableCOSI`
Trident 컨테이너 오브젝트 스토리지 인터페이스(COSI) 프런트엔드를 활성화합니다. `enableConcurrency` 을(를) `true`(으)로 설정해야 합니다. Trident COSI는 커뮤니티 COSI 컨트롤러와 함께 실행할 수 없습니다. 자세한 내용은 "COSI를 사용하여 오브젝트 스토리지 프로비저닝"을(를) 참조하십시오.	거짓입니다	`resources`

매개 변수	설명	기본값

Trident 컨트롤러와 노드 포드에 대한 Kubernetes 리소스 제한과 요청을 설정합니다. Kubernetes에서 리소스 할당을 관리하기 위해 각 컨테이너와 사이드카에 대한 CPU와 메모리를 구성할 수 있습니다.
 	
리소스 요청 및 제한 구성에 대한 자세한 내용은 다음을 참조하세요."포드 및 컨테이너에 대한 리소스 관리" .
 	
• 컨테이너나 필드의 이름을 변경하지 마세요.
• 들여쓰기를 변경하지 마세요. YAML 들여쓰기는 올바른 구문 분석을 위해 중요합니다.
 	
• 기본적으로 제한은 적용되지 않습니다. 요청에만 기본값이 있으며, 지정하지
	<pre>resources:
	  controller:
	    trident-main:
	      requests:
	        cpu: 10m
	        memory: 80Mi
	      limits:
	          cpu:
	          memory:
	    csi-provisioner:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-attacher:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-resizer:
	      requests:
	        cpu: 3m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-snapshotter:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    trident-autosupport:
	      requests:
	        cpu: 1m
	        memory: 30Mi
	      limits:
	        cpu:
	        memory:
	  node:</pre>
		httpsMetrics

매개 변수	설명	기본값
Prometheus 메트릭 엔드포인트에 HTTPS를 활성화합니다.	거짓	호스트 네트워크

POD 매개 변수 포맷에 대한 자세한 내용은 을 참조하십시오 "노드에 Pod 할당".

샘플 구성

속성을 사용할 수 있습니다구성 옵션 정의할 때 `TridentOrchestrator` 사용자 정의 설치를 위해.

▼ 기본 사용자 정의 구성

이 예제는 다음을 실행한 후에 생성됩니다. `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` 명령은 기본적인 사용자 정의 설치를 나타냅니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

▼ 노드 선택기

이 예제에서는 노드 선택기를 사용하여 Trident 설치합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

▼ **Windows** 작업자 노드

이 예제는 다음을 실행한 후에 생성됩니다. `cat deploy/crds/tridentorchestrator_cr.yaml` 명령을 실행하면 Windows 작업자 노드에 Trident 설치됩니다.

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

▼ **AKS** 클러스터의 관리 **ID**

이 예제에서는 AKS 클러스터에서 관리되는 ID를 활성화하기 위해 Trident 설치합니다.

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

▼ **AKS** 클러스터의 클라우드 **ID**

이 예제에서는 AKS 클러스터에서 클라우드 ID와 함께 사용할 Trident 설치합니다.

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

▼ **EKS** 클러스터의 클라우드 **ID**

이 예제에서는 AKS 클러스터에서 클라우드 ID와 함께 사용할 Trident 설치합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
```

```
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

▼ **GKE용 클라우드 ID**

이 예제에서는 GKE 클러스터에서 클라우드 ID와 함께 사용할 Trident 설치합니다.

```
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
```

▼ **Trident 컨트롤러 및 Trident Linux 노드 포드에 대한 Kubernetes 리소스 요청 및 제한 구성**

이 예제에서는 Trident 컨트롤러와 Trident Linux 노드 포드에 대한 Kubernetes 리소스 요청 및 제한을 구성합니다.



면책 조항: 이 예제에서 제공된 요청 및 제한 값은 데모 목적으로만 사용됩니다. 환경과 작업 부하 요구 사항에 따라 이러한 값을 조정하세요.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
```

```
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
```

```
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

▼ **Trident** 컨트롤러와 **Trident Windows** 및 **Linux** 노드 포드에 대한 **Kubernetes** 리소스 요청 및 제한 구성

이 예제에서는 Trident 컨트롤러와 Trident Windows 및 Linux 노드 포드에 대한 Kubernetes 리소스 요청 및 제한을 구성합니다.



면책 조항: 이 예제에서 제공된 요청 및 제한 값은 데모 목적으로만 사용됩니다. 환경과 작업 부하 요구 사항에 따라 이러한 값을 조정하세요.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
```

```
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
```

```
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

tridentctl을 사용하여 설치합니다

tridentctl을 사용하여 Trident를 설치합니다

을 사용하여 Trident를 설치할 수 `tridentctl` 있습니다. 이 프로세스는 Trident에 필요한 컨테이너 이미지가 개인 레지스트리에 저장되어 있거나 저장되지 않은 설치에 적용됩니다. 배포를 사용자 지정하려면 `tridentctl` 을 "tridentctl 배포를 사용자 지정합니다" 참조하십시오.

Trident 26.10에 대한 중요 정보

Trident에 대한 다음 중요 정보를 읽어야 합니다.

▼ ** Trident **에 대한 중요 정보입니다

- Kubernetes 1.37은 Trident 26.10에서 지원됩니다. Kubernetes를 업그레이드하기 전에 Trident를 업그레이드하십시오.
- Trident은 multipath.conf 파일에서 권장 값을 값으로 하여 SAN 환경에서 다중 경로 구성을 엄격히 적용합니다 `find_multipaths: no`.

  비 경로 다중화 구성 또는 의 사용 `find_multipaths: yes` 또는 `find_multipaths: smart` multipath.conf 파일의 값으로 인해 마운트 오류가 발생합니다. Trident에서 의 사용을 권장했습니다 `find_multipaths: no` 21.07 릴리스 이후.

를 사용하여 **Trident**를 설치합니다 `tridentctl`

"설치 개요"을(를) 검토하여 설치 사전 요구 사항을 충족했는지, 그리고 사용 환경에 맞는 올바른 설치 옵션을 선택했는지 확인하십시오.

시작하기 전에

설치를 시작하기 전에 Linux 호스트에 로그인하여 작동 중인 "지원되는 Kubernetes 클러스터"를 관리하고 있는지, 필요한 권한이 있는지 확인합니다.

> OpenShift에서는 뒤에 나오는 모든 예시에서 kubtl 대신 OC를 사용하고, OC login-u system:admin 또는 OC login-u kubbe-admin을 실행하여 먼저 * system:admin * 으로 로그인합니다.

▼ Details

1. Kubernetes 버전 확인:

```
kubectl version
```

2. 클러스터 관리자 권한 확인:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Docker Hub의 이미지를 사용하는 Pod를 시작하고 Pod 네트워크를 통해 스토리지 시스템에 연결할 수 있는지 확인합니다.

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

1단계**: Trident** 설치 프로그램 패키지를 다운로드합니다

Trident 설치 프로그램 패키지는 Trident POD를 생성하고, 상태를 유지하는 데 사용되는 CRD 객체를 구성하고, 클러스터 호스트에 볼륨을 프로비저닝하고 연결하는 등의 작업을 수행하기 위해 CSI 사이드카를 초기화합니다. 에서 최신 버전의 Trident 설치 프로그램을 다운로드하고 압축을 "GitHub의 _Assets_섹션"풉니다. 선택한 Trident 버전으로 예시에서 _<Trident-installer-XX.X.tar.gz>_를 업데이트하십시오.

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2단계**: Trident**를 설치합니다

명령을 실행하여 원하는 네임스페이스에 Trident를 `tridentctl install` 설치합니다. 추가 인수를 추가하여 이미지 레지스트리 위치를 지정할 수 있습니다.

표준 모드

```
./tridentctl install -n trident
```

하나의 레지스트리에 있는 이미지

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.10 --trident
-image <your-registry>/trident:26.10.1
```

다른 레지스트리의 이미지

```
./tridentctl install -n trident --image-registry <csi-registry>
--autosupport-image <autosupport-registry>/trident-autosupport:26.10
--trident-image <trident-registry>/trident:26.10.1
```

설치 상태는 다음과 같습니다.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=26.10.1
INFO Trident installation succeeded.
....
```

설치를 확인합니다

POD 생성 상태 또는 를 사용하여 설치를 확인할 수 있습니다 `tridentctl`.

POD 생성 상태 사용

생성된 Pod의 상태를 검토하여 Trident 설치가 완료되었는지 확인할 수 있습니다.

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

설치 프로그램이 성공적으로 완료되지 않거나 `trident-controller-<generated id>` 실행 중 상태가 아니면 플랫폼이 설치되지 않은 것입니다. `-d`을(를) 사용하여 "디버그 모드를 켭니다"하고 문제를 해결하십시오.

사용 **`tridentctl`**

를 사용하여 설치된 Trident 버전을 확인할 수 `tridentctl` 있습니다.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

샘플 구성

다음 예에서는 를 사용하여 Trident를 설치하기 위한 샘플 구성을 `tridentctl` 제공합니다.

▼ **Windows** 노드

Windows 노드에서 Trident를 실행하려면 다음을 따르십시오.

```
tridentctl install --windows -n trident
```

▼ 강제 분리

자세한 내용은 다음을 참조하세요."Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화" .

```
tridentctl install --enable-force-detach=true -n trident
```

▼ 동시 **Trident** 컨트롤러 작업 활성화

처리량 향상을 위해 Trident 컨트롤러의 동시 작업을 활성화하려면 이 예시와 같이 설치 중에 `--enable-concurrency` 옵션을 추가하십시오. 지원되는 드라이버에 대한 자세한 내용은 링크: Controller Scalability documentation를 참조하십시오.

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

▼ **NVMe** 데이터 **LIF** 새로 고침

LIF 구성이 변경될 때 게시된 볼륨의 NVMe 데이터 LIF를 새로 고치려면 `--enable-datalif-refresh`을(를) 사용하여 Trident를 설치하십시오. 지원되는 드라이버 및 구성 세부 정보는 "NVMe 데이터 LIF 새로 고침"을(를) 참조하십시오.

```
tridentctl install --enable-datalif-refresh -n trident
```

Trident tridentctl 설치 사용자 정의

Trident 설치 프로그램을 사용하여 설치를 사용자 지정할 수 있습니다.

설치 프로그램에 대해 알아보십시오

Trident 설치 프로그램을 사용하여 특성을 사용자 지정할 수 있습니다. 예를 들어 Trident 이미지를 개인 리포지토리에 복사한 경우 을 사용하여 이미지 이름을 지정할 수 `--trident-image` 있습니다. 필요한 CSI 사이드카 이미지와 Trident 이미지를 개인 리포지토리에 복사한 경우 양식을 사용하는 스위치를 `<registry FQDN>[:port]` 사용하여 리포지토리의 위치를 지정하는 것이 좋습니다 `--image-registry`.

> 전용 리포지토리에 Trident를 설치할 때 스위치를 사용하여 리포지토리 위치를 지정하는 경우 `--image-registry` 리포지토리 경로에서 를 사용하지 마십시오 `/netapp/`. 예를 들면 다음과 같습니다.
> `./tridentctl install --image-registry <image-registry> -n <namespace>`

kubelet이 평소 하던 /var/lib/kubelet가 아닌 다른 경로에 데이터를 보관하는 Kubernetes 배포를 사용하는 경우 -kubelet-dir을 사용하여 대체 경로를 지정할 수 있습니다.

설치 관리자의 인수 이외에 설치를 사용자 지정해야 하는 경우 배포 파일을 사용자 지정할 수도 있습니다. '--generate-custom-YAML' 파라미터를 사용하면 설치 프로그램의 '셋업' 디렉토리에 다음과 같은 YAML 파일이 생성됩니다.

- 트리덴트-클러스터롤리바인딩.YAML
- '트라이덴트 배포.YAML'
- 트리덴트-CRD.YAML
- 트리덴트-클러스터역할.YAML
- 트리덴트-디멘트발병 YAML
- '삼류 서비스.YAML'
- '삼중인대.YAML'
- 트라이덴트-서비스계정.YAML
- `trident-resourcequota.yaml`*

이러한 파일을 생성한 후 필요에 따라 수정한 다음 '--use-custom-YAML'을 사용하여 사용자 지정 배포를 설치할 수 있습니다.

```
./tridentctl install -n trident --use-custom-yaml
```

OpenShift 인증 운영자를 사용하여 설치합니다

OpenShift OperatorHub를 사용하여 Trident를 설치합니다

Red Hat OpenShift를 사용하는 경우 Red Hat 공인 운영자를 사용하여 NetApp Trident를 설치할 수 있습니다. 다음 절차를 사용하여 Red Hat OpenShift Container Platform에서 Trident를 설치합니다.

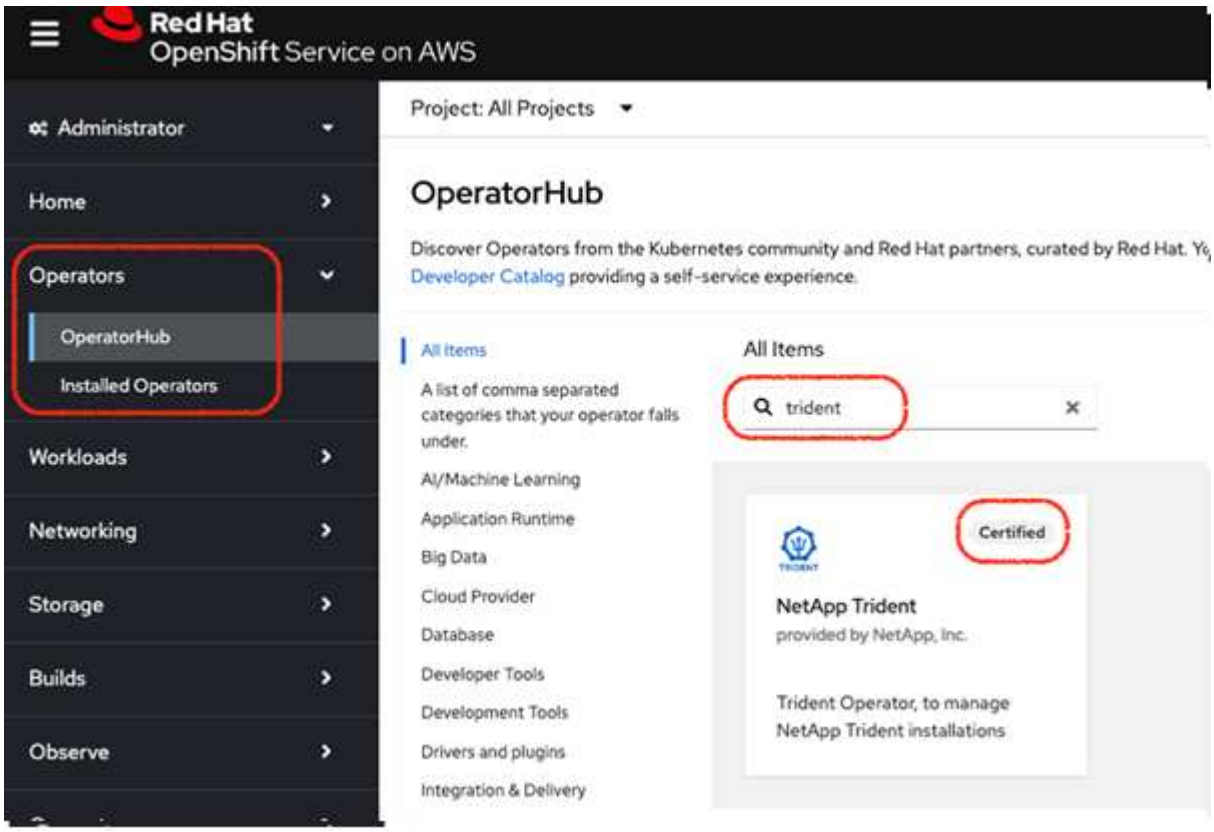
시작하기 전에

설치를 시작하기 전에 "Trident 설치 환경을 준비합니다"를 참조하십시오.

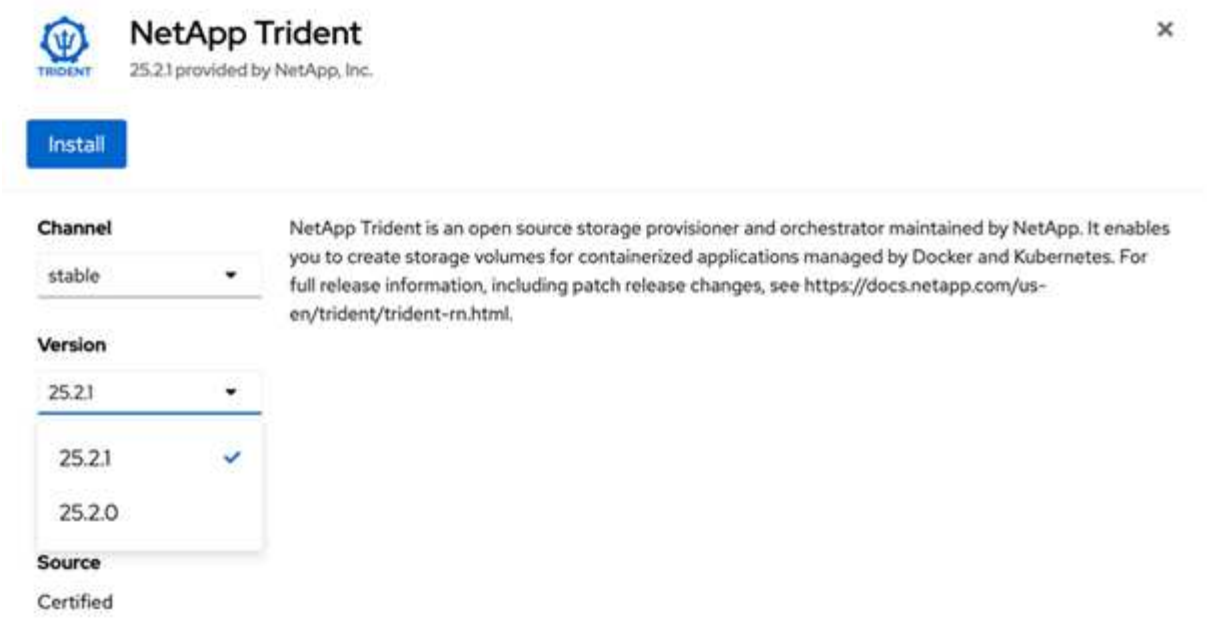
Trident 운영자를 찾아 설치합니다

단계

1. OpenShift OperatorHub로 이동하여 NetApp Trident를 검색합니다.

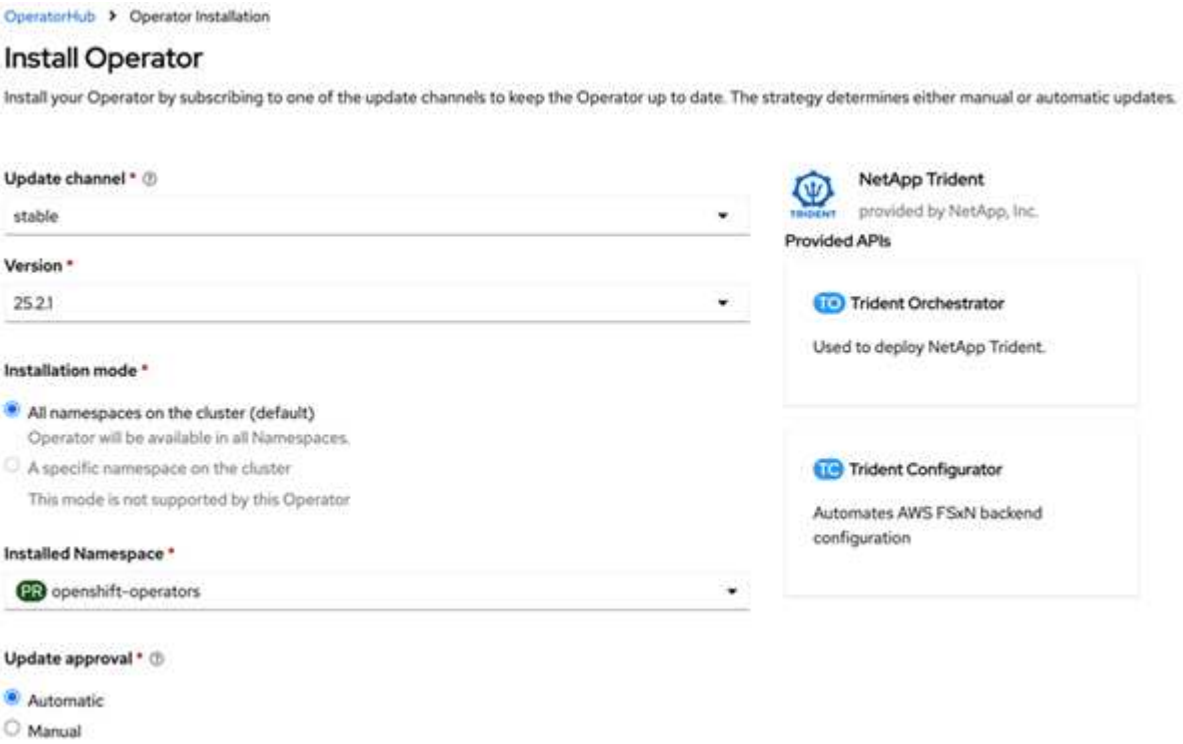


2. * NetApp Trident*를 클릭하여 설치 설정을 엽니다.
3. 필요한 옵션을 선택하고 *설치*를 클릭하여 운영자 구성을 엽니다.

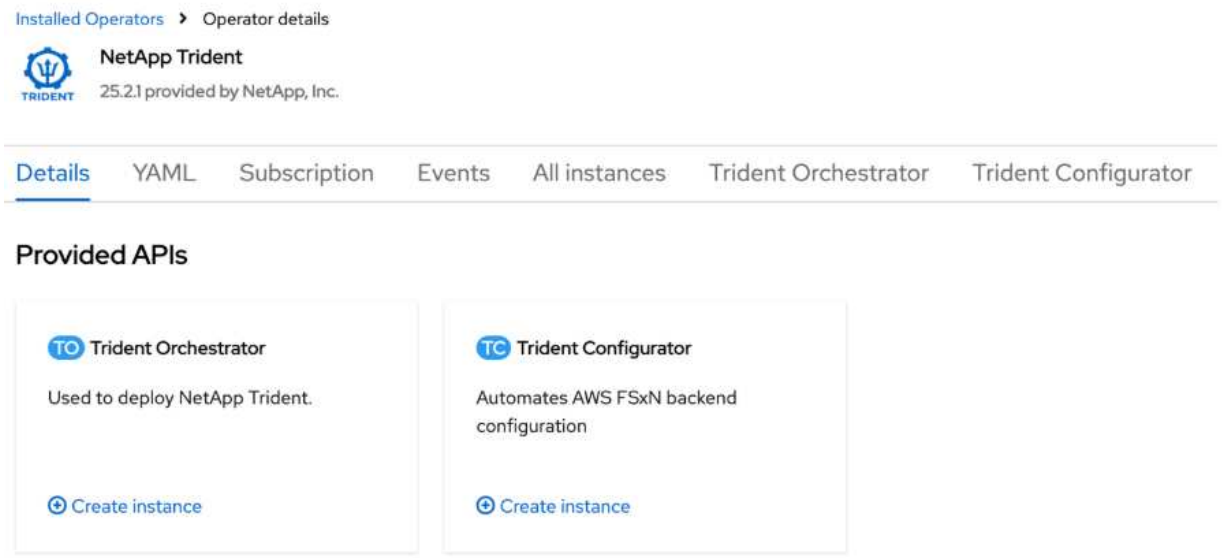


최신 운영자 버전을 선택했는지 확인하세요.

4. 모든 매개변수를 그대로 두고 *설치*를 클릭합니다.



5. 운영자 세부 정보를 보려면 *운영자 보기*를 클릭하세요.



6. *YAML 보기*를 클릭하고 다음을 양식에 붙여넣습니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

사용자 인터페이스는 기본 예시를 제공합니다. 전체 구성을 복사하는 대신 직접 편집할 수 있습니다.

선택 사항: 동시 실행 활성화

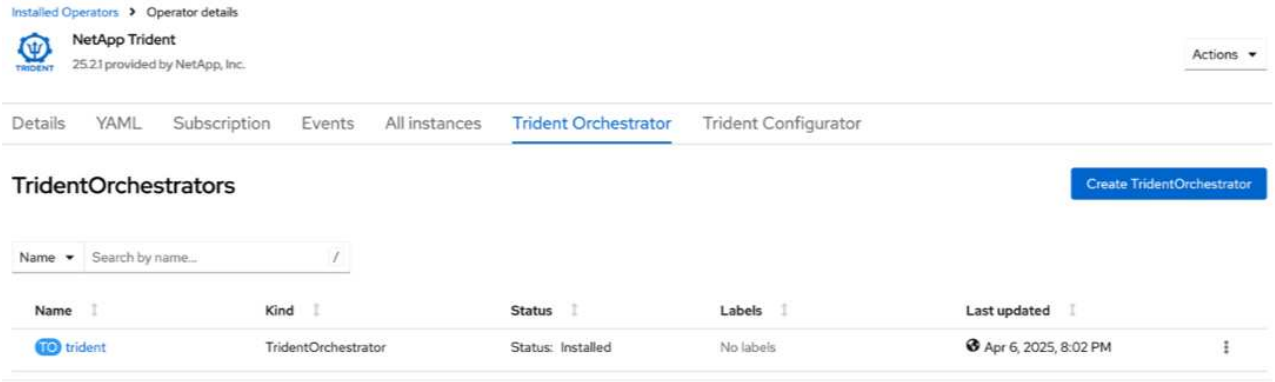
동시성을 활성화하려면 spec에 다음 필드를 추가하십시오.

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS(RHCOS)에는 iSCSI가 활성화 및 구성되어 있지 않습니다.
- 추가할 수 있습니다 `nodePrep` 모든 OpenShift 워커 노드에서 iSCSI 및 Multipath 서비스를 구성하고 활성화하기 위한 매개변수입니다.

- OpenShift 4.19부터 이 기능을 지원하는 최소 Trident 버전은 25.06.1입니다.

1. *만들기*를 클릭하면 Trident Orchestrator가 완전히 설치됩니다.



Trident 운영자 제거

단계

1. 설치된 운영자 목록에서 Trident 운영자를 선택합니다.
2. 연산자에서 모든 피연산자 인스턴스를 삭제하려면 선택합니다.

> 이 연산자에서 모든 피연산자 인스턴스 삭제 * 확인란을 선택하지 않으면 Trident가 제거되지 않습니다.

3. 제거 * 를 클릭합니다.

OpenShift 인증 Trident 운영자로 전환하세요

Red Hat OpenShift 인증 Trident 오퍼레이터는 커뮤니티 오퍼레이터, Helm 기반 설치 또는 수동으로 배포된 오퍼레이터에서 전환할 수 있습니다. 각 방법의 프로세스에는 기존 오퍼레이터를 제거한 다음 OperatorHub를 사용하여 인증된 오퍼레이터를 설치하는 작업이 포함됩니다.

시작하기 전에

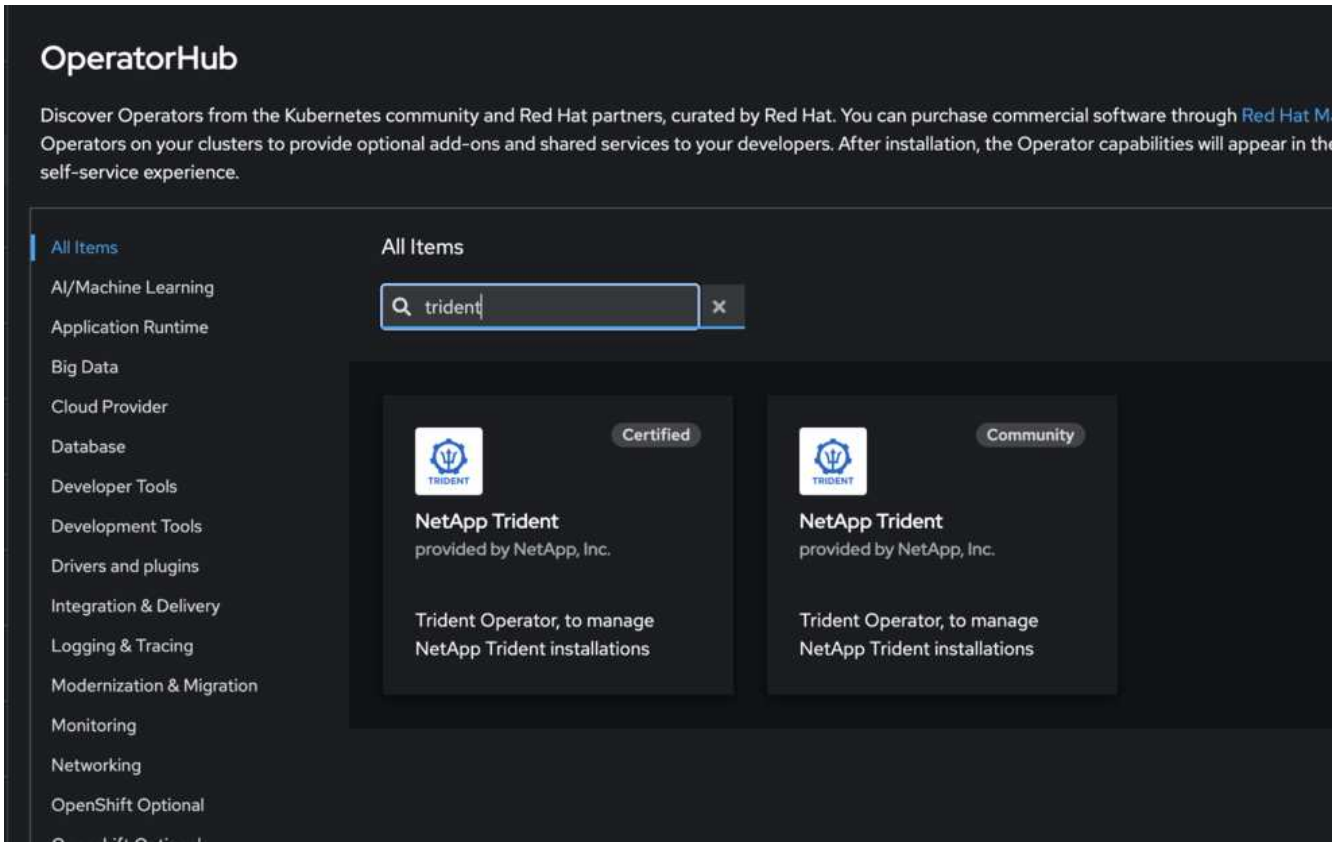
설치를 시작하기 전에 "Trident 설치 환경을 준비합니다"를 참조하십시오.

> `TridentOrchestrator` 사용자 지정 리소스(CR)를 제거 과정 중에 삭제하지 마십시오. `TridentOrchestrator` CR은 인증된 운영자를 설치한 후 필요한 백엔드 및 스토리지 클래스 구성을 유지합니다.

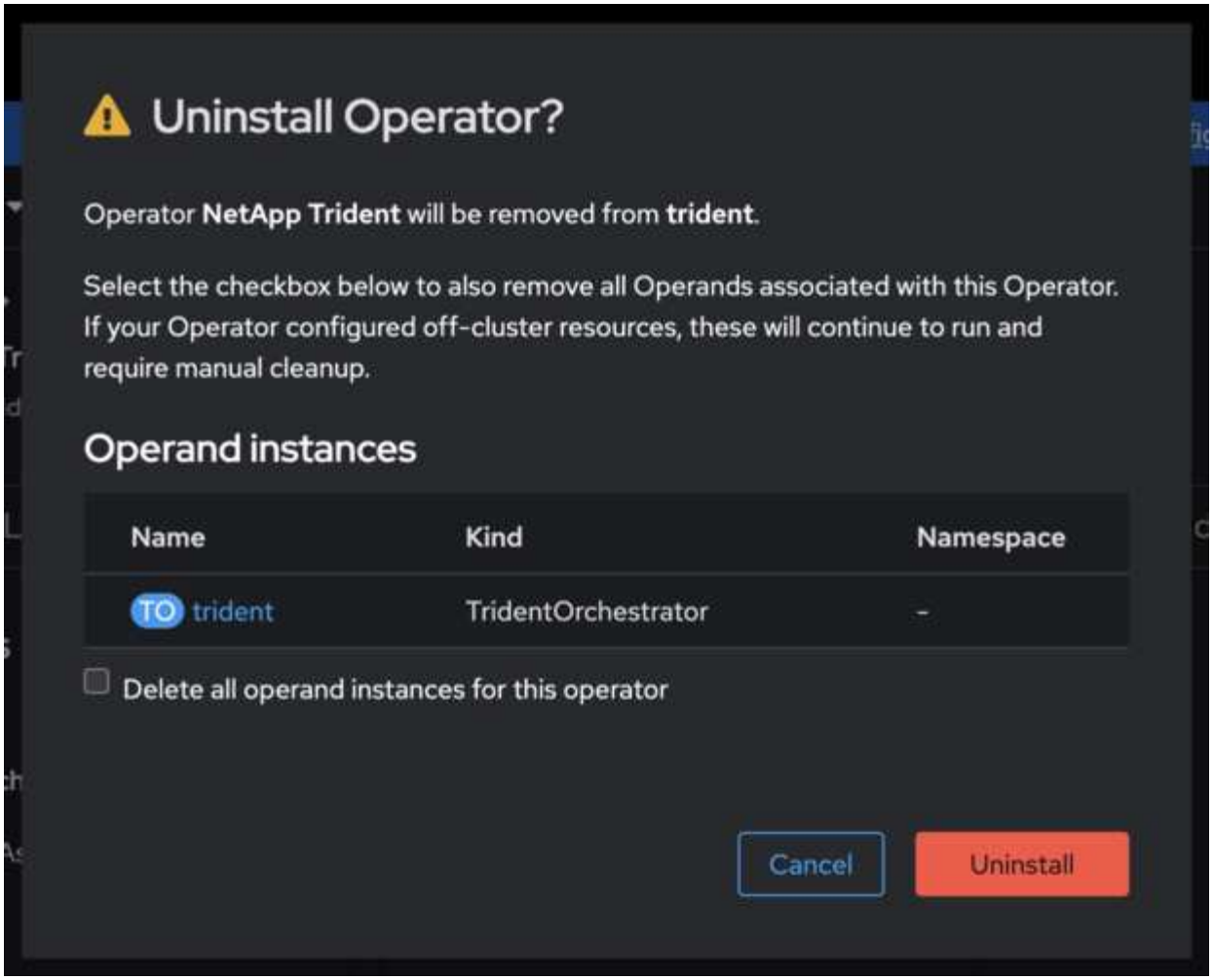
커뮤니티 운영자에서 전환

단계

1. OpenShift 콘솔을 사용하여 OperatorHub로 이동합니다.



2. NetApp Trident 커뮤니티 운영자를 찾습니다.



이 연산자에서 모든 피연산자 인스턴스 삭제 * 를 선택하지 마십시오.

3. 제거 * 를 클릭합니다.
4. 제거가 완료되면 OpenShift 인증 운영자를 설치하세요(으)로 진행하십시오.

Helm 기반 운영자 설치에서 전환

단계

1. Trident 설치에 대한 Helm 릴리스를 나열합니다.

```
helm ls -n trident
```

2. Helm 릴리스를 제거하세요:

```
helm uninstall <release-name> -n trident
```

3. 제거가 완료되면 OpenShift 인증 운영자를 설치하세요(으)로 진행하십시오.

수동으로 배포된 **operator**에서 전환

설치 패키지의 `bundle.yaml`를 사용하여 수동으로 운영자를 배포하여 Trident를 설치한 경우 동일한 매니페스트를 삭제하여 제거하십시오.

단계

1. 번들 매니페스트를 사용하여 운영자 배포를 삭제합니다.

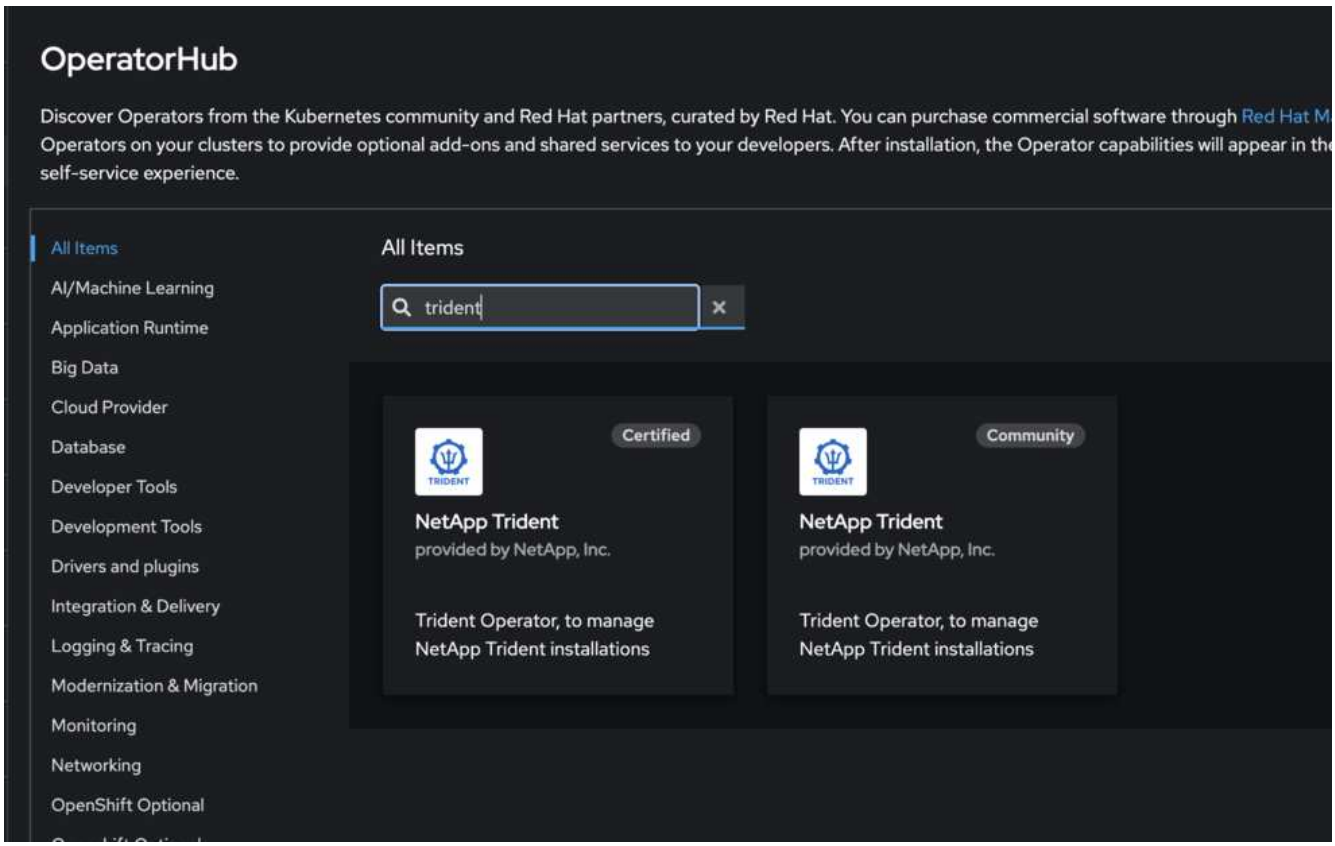
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. 제거가 완료되면 OpenShift 인증 운영자를 설치하세요(으)로 진행하십시오.

OpenShift 인증 운영자를 설치하세요

단계

1. Red Hat OperatorHub 로 이동합니다.
2. NetApp Trident 연산자를 검색하여 선택합니다.



3. 화면의 지침에 따라 작업자를 설치합니다.

검증

- 콘솔에서 OperatorHub 를 확인하여 새 인증 운영자가 성공적으로 설치되었는지 확인합니다.

Trident를 사용합니다

Trident 워커 노드를 준비합니다

Kubernetes 클러스터의 모든 작업자 노드는 Pod용으로 프로비저닝된 볼륨을 마운트할 수 있어야 합니다. 작업자 노드를 준비하려면 드라이버 선택에 따라 NFS, iSCSI, NVMe/TCP 또는 FC 툴을 설치해야 합니다.

올바른 도구 선택

드라이버 조합을 사용하는 경우 드라이버에 필요한 모든 도구를 설치해야 합니다. 최신 버전의 Red Hat Enterprise Linux CoreOS(RHCOS)에는 기본적으로 도구가 설치되어 있습니다.

NFS 툴

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

iSCSI 툴

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

NVMe 툴

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

NetApp은 NVMe/TCP에 ONTAP 9.12 이상을 권장합니다.

SCSI over FC 도구

자세한 내용은 을 link:https://docs.netapp.com/us-en/ontap/san-config/configure-fc-nvme-hosts-ha-pairs-reference.html["FC 및 AMP, FC-NVMe SAN 호스트를 구성하는 방법"]FC 및 FC-NVMe SAN 호스트 구성에 대한참조하십시오.

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

- 고려해야 할 점 *: * OpenShift 및 KubeVirt 환경에서 FC 기반 SCSI가 지원됩니다. *SCSI over FC는 Docker에서 지원되지 않습니다. *iSCSI 자동 복구는 FC를 통한 SCSI에는 적용되지 않습니다.

SMB 도구

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

노드 서비스 검색

Trident는 노드에서 iSCSI 또는 NFS 서비스를 실행할 수 있는지 여부를 자동으로 감지합니다.

노드 서비스 검색은 검색된 서비스를 식별하지만 서비스가 올바르게 구성된다고 보장하지 않습니다. 반대로 검색된 서비스가 없으면 볼륨 마운트가 실패한다고 보장할 수 없습니다.

이벤트를 검토합니다

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 서비스를 검토합니다

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

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

NFS 볼륨

운영 체제의 명령을 사용하여 NFS 툴을 설치합니다. 부팅 중에 NFS 서비스가 시작되었는지 확인합니다.

RHEL 8+

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

우분투

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



볼륨에 연결할 때 오류가 발생하지 않도록 NFS 툴을 설치한 후 작업자 노드를 재부팅합니다.

iSCSI 볼륨

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

iSCSI 자동 복구 기능

ONTAP 시스템의 경우 Trident는 5분마다 iSCSI 자동 복구를 실행하여 다음을 수행합니다.

1. * 원하는 iSCSI 세션 상태와 현재 iSCSI 세션 상태를 식별합니다.
2. * 원하는 상태를 현재 상태와 비교 * 하여 필요한 수리를 확인합니다. Trident는 수리 우선 순위와 수리 선점 시기를 결정합니다.
3. * 현재 iSCSI 세션 상태를 원하는 iSCSI 세션 상태로 되돌리는 데 필요한 복구 수행 *



자가 치유 활동 로그는 `trident-main` 해당 Demonset POD의 컨테이너에 있습니다. 로그를 보려면

> Trident 설치 중에 "true"로 설정해야 `debug` 합니다.

Trident iSCSI 자동 복구 기능은 다음과 같은 이점을 제공합니다.

- 네트워크 연결 문제가 발생한 후 발생할 수 있는 오래되거나 비정상적인 iSCSI 세션. 오래된 세션의 경우 Trident는 로그아웃하기 전에 7분 동안 기다렸다가 포털과의 연결을 다시 설정합니다.

> ⓘ 예를 들어, 스토리지 컨트롤러에서 CHAP 암호를 회전시키고 네트워크에서 연결이 끊어지면 이전의 (*stale*) CHAP 암호가 지속될 수 있습니다. 자동 복구 기능은 이 문제를 인식하고 업데이트된 CHAP 암호를 적용하기 위해 세션을 자동으로 다시 설정할 수 있습니다.

- iSCSI 세션이 누락되었습니다
- LUN이 없습니다
- Trident 업그레이드 전에 고려해야 할 사항 *
- 23.04+에서 도입된 노드별 igroup만 사용 중인 경우, iSCSI 자동 복구가 SCSI 버스의 모든 장치에 대해 SCSI 재검색을 시작합니다.
- 백엔드 범위 igroup(23.04부터 더 이상 사용되지 않음)만 사용 중인 경우 iSCSI 자동 복구가 SCSI 버스에서 정확한 LUN ID에 대한 SCSI 재검색을 시작합니다.
- 노드당 igroup과 백엔드 범위 igroup이 함께 사용되고 있는 경우 iSCSI 자동 복구가 SCSI 버스에서 정확한 LUN ID에 대한 SCSI 재검색을 시작합니다.

iSCSI 도구를 설치합니다

운영 체제의 명령을 사용하여 iSCSI 도구를 설치합니다.

시작하기 전에

- Kubernetes 클러스터의 각 노드에는 고유한 IQN이 있어야 합니다. * 이것은 필수 전제 조건입니다 *.
- RHCOS 버전 4.5 이상 또는 기타 RHEL 호환 Linux 배포를 사용하는 경우 를 참조하십시오 `solidfire-san` 드라이버 및 Element OS 12.5 이전 버전에서는 CHAP 인증 알고리즘이 에서 MD5로 설정되어 있는지 확인합니다 `/etc/iscsi/iscsid.conf`. 보안 FIPS 호환 CHAP 알고리즘 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
```

- iSCSI PVS와 함께 RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 사용하는 경우 StorageClass에서 mount옵션을 지정하여 `discard` 인라인 공간 재확보를 수행합니다. 을 "Red Hat 설명서 "참조하십시오.
- 최신 버전으로 업그레이드했는지 확인하세요. `multipath-tools` .

RHEL 8+

1. 다음 시스템 패키지를 설치합니다.

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

```
multipath
```

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

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

3. 스캔을 수동으로 설정합니다.

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

4. 다중 경로 설정:

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

> `/etc/multipath.conf`아래 내용을 `defaults` 포함해야 `find_multipaths no` 합니다.

5. iscsid와 multipathd가 실행 중인지 확인합니다.

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

6. "iSCSI" 활성화 및 시작:

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

우분투

1. 다음 시스템 패키지를 설치합니다.

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

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic) 또는 2.0.874-7.1uubuttu6.1 이상(focal)인지 확인합니다.

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

3. 스캔을 수동으로 설정합니다.

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

4. 다중 경로 설정:

```
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`아래 내용을 `defaults 포함해야 `find_multipaths no` 합니다.

5. 'open-iscsi'와 'multirpath-tools'가 활성화되어 실행되고 있는지 확인합니다.

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

> Ubuntu 18.04의 경우 iSCSI 데몬이 시작되도록 "open-iscsi"를 시작하기 전에 iscsiadm"이 있는 타겟 포트를 검색해야 합니다. 또는 iSCSI 서비스를 수정하여 iscsid를 자동으로 시작할 수 있습니다.

iSCSI 자동 복구를 구성하거나 사용하지 않도록 설정합니다

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

- **iscsi** 자동 복구 간격: iSCSI 자동 복구가 호출되는 빈도를 결정합니다(기본값: 5분). 더 큰 숫자를 설정하여 더 적은 숫자를 설정하거나 더 자주 실행되도록 구성할 수 있습니다.

> iSCSI 자동 복구 간격을 0으로 설정하면 iSCSI 자동 복구가 완전히 중지됩니다. iSCSI 자동 복구를 비활성화하는 것은 권장하지 않습니다. iSCSI 자동 복구가 의도된 대로 작동하지 않거나 디버깅 목적으로 작동하지 않는 특정 시나리오에서만 비활성화해야 합니다.

- * iSCSI 자동 복구 대기 시간 *: 비정상 세션에서 로그아웃하고 다시 로그인을 시도하기 전에 iSCSI 자동 복구 대기 시간을 결정합니다(기본값: 7분). 상태가 좋지 않은 것으로 확인된 세션이 로그아웃되기 전에 더 오래 대기해야 하고 다시 로그인하려고 시도하거나 더 적은 수의 숫자를 사용하여 이전에 로그아웃하도록 구성할 수 있습니다.

헬름

iSCSI 자동 복구 설정을 구성하거나 변경하려면 를 전달합니다 `iscsiSelfHealingInterval` 및 `iscsiSelfHealingWaitTime` Helm 설치 또는 Helm 업데이트 중 매개변수.

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

```
helm install trident trident-operator-100.2610.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl 을 선택합니다

iSCSI 자동 복구 설정을 구성하거나 변경하려면 를 전달합니다 `iscsi-self-healing-interval` 및 `iscsi-self-healing-wait-time` tridentctl 설치 또는 업데이트 중 매개 변수입니다.

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

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

NVMe/TCP 볼륨

운영 체제의 명령을 사용하여 NVMe 툴을 설치합니다.

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

RHEL 9 를 참조하십시오

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

우분투

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

설치를 확인합니다

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

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

Trident는 이 값을 수정하여 `ctrl_device_tmo` NVMe가 다운될 경우 경로를 포기하지 않도록 합니다. 이 설정을 변경하지 마십시오.

FC 볼륨을 통한 SCSI

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

필수 구성 요소

FC에 필요한 네트워크 및 노드 설정을 구성합니다.

네트워크 설정

1. 대상 인터페이스의 WWPN을 가져옵니다. 자세한 내용은 을 "네트워크 인터페이스가 표시됩니다" 참조하십시오.
2. 이니시에이터(호스트)의 인터페이스에 대한 WWPN을 가져옵니다.

   해당 호스트 운영 체제 유틸리티를 참조하십시오.

3. 호스트와 대상의 WWPN을 사용하여 FC 스위치에서 조닝을 구성합니다.

   자세한 내용은 존경하는 스위치 공급업체 문서를 참조하십시오.

   자세한 내용은 다음 ONTAP 설명서를 참조하십시오.

   - "파이버 채널 및 FCoE 조닝 개요"
   - "FC 및 AMP, FC-NVMe SAN 호스트를 구성하는 방법"

FC 툴을 설치합니다

운영 체제의 명령을 사용하여 FC 툴을 설치합니다.

- RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 FC PVS와 함께 사용하는 경우 StorageClass에서 mount옵션을 지정하여 `discard` 인라인 공간 재확보를 수행합니다. 을 "Red Hat 설명서 "참조하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치합니다.

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

2. 다중 경로 설정:

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

> `/etc/multipath.conf`아래 내용을 `defaults` 포함해야 `find_multipaths no` 합니다.

3. 가 `multipathd` 실행 중인지 확인합니다.

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

우분투

1. 다음 시스템 패키지를 설치합니다.

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

2. 다중 경로 설정:

```
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`아래 내용을 `defaults` 포함해야 `find_multipaths no` 합니다.

3. 가 활성화되어 있고 실행 중인지 `multipath-tools` 확인합니다.

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

SMB 볼륨 프로비저닝을 위한 준비

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

> ONTAP 온프레미스 클러스터를 위한 SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 `ontap-nas-economy` 합니다. 이 두 프로토콜 중 하나를 구성하지 않으면

SMB 볼륨 생성에 실패합니다.

`autoExportPolicy` SMB 볼륨에는 가 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하려면 먼저 다음 항목이 있어야 합니다.

- Linux 컨트롤러 노드 및 Windows Server 2022를 실행하는 Windows 작업자 노드가 있는 Kubernetes 클러스터 Trident는 Windows 노드에서만 실행되는 Pod에 마운트된 SMB 볼륨을 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 암호가 하나 이상 있습니다. 비밀 생성하기 `smbcreds`:

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

- Windows 서비스로 구성된 CSI 프록시. 를 구성합니다 `csi-proxy`를 참조하십시오 "GitHub:CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우:

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident에서 공유를 생성할 수 있습니다.

   ONTAP용 Amazon FSx에는 SMB 공유가 필요합니다.

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

   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 생성합니다.

      를 클릭합니다 `vserver cifs share create` 명령은 공유를 생성하는 동안 -path 옵션에 지정된 경로를 확인합니다. 지정한 경로가 없으면 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
```

   을 참조하십시오 "SMB 공유를 생성합니다" 를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 을 참조하십시오 "ONTAP 구성 옵션 및 예제용 FSX".

매개 변수	설명	예
`smbShare`	Microsoft 관리 콘솔 또는 ONTAP CLI를 사용하여 생성된 SMB 공유의 이름, Trident에서 SMB 공유를 생성할 수 있는 이름, 볼륨에 대한 일반적인 공유 액세스를 방지하기 위해 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 사내 ONTAP의 경우 선택 사항입니다. 이 매개변수는 ONTAP 백엔드에 대한 아마존 FSx에 필요하며 비워둘 수 없습니다.	`smb-share`
`nasType`	* 를 로 설정해야 합니다 `smb`. * null인 경우 기본값은 로 설정됩니다 `nfs`.	`smb`
'생태성 스타일'을 참조하십시오	새로운 볼륨에 대한 보안 스타일 * 를 로 설정해야 합니다 `ntfs` 또는 `mixed` SMB 볼륨용. *	`ntfs` 또는 `mixed` SMB 볼륨용
유니크권한	모드를 선택합니다. SMB 볼륨에 대해서는 * 를 비워 두어야 합니다. *	""

COSI를 사용하여 오브젝트 스토리지 프로비저닝

Container Object Storage Interface(COSI)를 사용하여 Trident가 관리하는 Kubernetes 리소스를 통해 Kubernetes 워크로드용 S3 버킷을 프로비저닝하십시오.

개요

COSI는 오브젝트 스토리지 공급자를 Kubernetes 워크로드에 연결합니다. Trident는 다음과 같은 리소스를 제공합니다.

- `TridentObjectStore`: Trident를 오브젝트 스토리지 시스템 및 해당 자격 증명에 연결합니다. 이 리소스를 Trident 네임스페이스에 생성합니다.
- `BucketClass`: 오브젝트 저장소 선택 및 버킷 삭제 정책을 정의합니다.
- `BucketClaim`: 버킷을 요청합니다.
- `BucketAccessClass`: 버킷에 대한 액세스 권한이 어떻게 제공되는지 정의합니다.
- `BucketAccess`: 하나 이상의 버킷에 대한 자격 증명을 요청합니다.

Trident는 다음과 같은 오브젝트 스토리지 드라이버를 사용하여 COSI를 지원합니다:

- StorageGRID: 정식 출시되었습니다.
- ONTAP S3: 기술 미리보기.

Trident는 S3 프로토콜을 지원합니다. 초기 버전에서는 경로 기반 S3 주소 지정 방식만 지원합니다.

요구 사항 및 제한 사항

- `enableConcurrency`와 `enableCOSI` 모두를 `TridentOrchestrator` 사용자 정의 리소스(TORC)에서 활성화합니다. COSI는 기본적으로 활성화되어 있지 않습니다.
- 컨트롤러 동시성을 활성화하기 전에 구성된 모든 Trident 백엔드가 지원되는 드라이버를 사용하는지 확인하십시오. 지원되지 않는 백엔드 드라이버를 사용하면 Trident가 시작되지 않습니다. "컨트롤러 확장성을 활성화하기 전에"을

(를) 참조하십시오.

- Trident COSI를 커뮤니티 COSI 컨트롤러와 함께 실행하지 마십시오.
- 초기 릴리스에서는 `objectstorage.k8s.io/v1alpha2`COSI API를 사용합니다.
- StorageGRID `TridentObjectStore` 리소스는 장기간 유효한 테넌트 루트 자격 증명을 사용합니다. 각각 `TridentObjectStore`은 하나의 StorageGRID 테넌트에 연결되며, 여러 리소스가 동일한 테넌트에 연결될 수 있습니다. Trident 네임스페이스의 Kubernetes Secret에 자격 증명을 저장하고 보안 요구 사항에 따라 보호하십시오.
- StorageGRID `BucketAccess`는 최대 8개의 버킷을 지원합니다.

동시 컨트롤러 작업에 대한 일반적인 정보는 "컨트롤러 확장성"를 참조하십시오.

COSI 활성화

`TridentOrchestrator` 사양에서 두 옵션을 모두 설정하십시오:

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

일반적인 운영자 워크플로를 사용하여 TORC를 적용하거나 기존 리소스를 업데이트하십시오. TORC 구성에 대한 자세한 내용은 "Trident 운영자 설치를 사용자 지정합니다"를 참조하십시오.

StorageGRID 오브젝트 저장소를 구성합니다.

Trident 네임스페이스에 StorageGRID 테넌트 자격 증명으로 Kubernetes Secret을 생성합니다:

```
apiVersion: v1
kind: Secret
metadata:
  name: tenant3
  namespace: trident
type: Opaque
stringData:
  username: <tenant-root-username>
  password: <tenant-root-password>
```

Secret을 참조하는 `TridentObjectStore`를 생성합니다. 예시 엔드포인트와 계정 ID를 StorageGRID 환경에 맞는 값으로 변경합니다:

```
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: trident-tenant3
  namespace: trident
spec:
  accountId: "<account-id>"
  credentials:
    secretName: tenant3
  labels:
    tenantInstance: three
  managementEndpoint: https://<storagegrid-management-endpoint>
  objectStoreDriver: storage-grid
  s3Endpoint: https://<storagegrid-s3-endpoint>
```

두 리소스를 모두 적용합니다:

```
kubectl apply -f tenant-secret.yaml
kubectl apply -f trident-object-store.yaml
```

ONTAP S3 오브젝트 저장소 구성

COSI를 통한 ONTAP S3 지원은 Trident 26.10의 기술 미리보기 기능이며, 실제 운영 환경에서는 지원되지 않습니다. 구성 필드 및 동작은 정식 출시 전에 변경될 수 있습니다.

Trident 네임스페이스에 ONTAP S3 사용자 자격 증명을 사용하여 Kubernetes Secret을 생성한 다음, 해당 Secret을 참조하는 `TridentObjectStore`를 생성합니다. 자리 표시자를 ONTAP S3 환경에 맞는 값으로 바꿉니다:

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-s3-credentials
  namespace: trident
type: Opaque
stringData:
  username: "<user>"
  password: "<password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: ontap-s3
  namespace: trident
```

```
spec:
  objectStoreDriver: ontap-s3
  objectStoreName: ontap-object-store
  managementEndpoint: "<management-endpoint>"
  svmName: "<svm-name>"
  storagePrefix: trident
  credentials:
    secretName: ontap-s3-credentials
```

`storagePrefix`은(는) 선택 사항이며 생략 시 기본값은 `trident`입니다. Secret 및 `TridentObjectStore`을(를) 적용합니다:

```
kubectl apply -f ontap-s3-object-store.yaml
```

BucketClass를 생성하고 버킷을 요청합니다

`BucketClass`은(는) 레이블 선택기, 오브젝트 저장소 유형 또는 둘 다를 사용하여 오브젝트 저장소를 선택할 수 있습니다. 선택 매개변수가 없으면 Trident는 구성된 모든 오브젝트 저장소 중에서 선택할 수 있습니다.

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClass
metadata:
  name: sg-delete
spec:
  driverName: cosi.trident.netapp.io
  deletionPolicy: Delete
  parameters:
    selector: tenantInstance=three
    objectStoreType: storage-grid
```

클래스를 사용하는 `BucketClaim`을(를) 생성하십시오:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClaim
metadata:
  name: claim-a1
  namespace: default
spec:
  bucketClassName: sg-delete
```

```
  protocols:
    - S3
```

Trident가 클레임을 처리할 때 일치하는 오브젝트 저장소를 선택합니다. 일치하는 저장소가 여러 개 있는 경우 Trident는 그중 하나를 무작위로 선택하여 일치하는 저장소 전체에 버킷을 분산합니다.

버킷에 워크로드 액세스 권한 부여

액세스 프로비저닝을 정의하는 `BucketAccessClass`을(를) 생성합니다:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccessClass
metadata:
  name: sg-access
spec:
  authenticationType: Key
  driverName: cosi.trident.netapp.io
  multiBucketAccess: MultipleBuckets
```

버킷 클레임을 참조하는 `BucketAccess`을(를) 생성합니다:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccess
metadata:
  name: sg-claim-a1
  namespace: default
spec:
  bucketAccessClassName: sg-access
  bucketClaims:
    - accessMode: ReadWrite
      accessSecretName: sg-claim-a1
      bucketClaimName: claim-a1
  protocol: S3
```

Trident는 버킷에 대한 액세스 정보가 포함된 Kubernetes Secret을 생성합니다. 워크로드는 표준 Kubernetes Secret 메커니즘을 사용하여 이 Secret을 사용할 수 있습니다. 생성된 Secret에는 버킷 식별자, 엔드포인트, 리전, 액세스 키 등 S3 연결 세부 정보가 포함됩니다. 생성된 Secret을 권한이 없는 사용자 또는 워크로드에 노출하지 마십시오.

생성된 Secret에는 `COSI_S3_ACCESS_KEY_ID`, `COSI_S3_ACCESS_SECRET_KEY`, `COSI_S3_ADDRESSING_STYLE`, `COSI_S3_BUCKET_ID`, `COSI_S3_ENDPOINT` 및 `COSI_S3_REGION` 등의 키가 포함됩니다.

백엔드 구성 및 관리

Trident 백엔드 구성

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

Trident는 스토리지 클래스에 정의된 요구 사항에 맞는 백 엔드에서 스토리지 풀을 자동으로 제공합니다. 스토리지 시스템에 대한 백엔드를 구성하는 방법에 대해 알아봅니다.

- "Azure NetApp Files 백엔드를 구성합니다"
- "Google Cloud NetApp 볼륨 백엔드를 구성합니다"
- "NetApp HCI 또는 SolidFire 백엔드를 구성합니다"
- "ONTAP 또는 Cloud Volumes ONTAP NAS 드라이버를 사용하여 백엔드를 구성합니다"
- "ONTAP 또는 Cloud Volumes ONTAP SAN 드라이버를 사용하여 백엔드를 구성합니다"
- "Trident를 Amazon FSx for NetApp ONTAP와 함께 사용해 보십시오"

Azure NetApp Files

Azure NetApp Files 백엔드를 구성합니다

Azure NetApp Files를 Trident의 백엔드로 사용합니다. 이 백엔드는 NFS 및 SMB 볼륨을 지원합니다. Trident는 Azure Kubernetes Service(AKS) 클러스터에 대한 관리 ID 및 워크로드 ID를 지원합니다.

지원되는 Azure 클라우드 환경

Trident는 다양한 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)
- Azure China

Trident를 배포하거나 Azure NetApp Files 백엔드를 구성할 때 Azure Resource Manager 및 인증 엔드포인트가 Azure 클라우드 환경과 일치하는지 확인하십시오.

Azure NetApp Files 드라이버 지원 검토

Trident는 다음과 같은 Azure NetApp Files 스토리지 드라이버를 제공합니다.

지원되는 액세스 모드에는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(RWX) 및 *ReadWriteOncePod*(RWOP)가 포함됩니다.

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'Azure-NetApp-파일'	NFS 를 참조하십시오 중소기업	파일 시스템	RWO, ROX, rwx, RWOP	`nfs, smb`

검토 시 고려 사항

- Azure NetApp Files는 50GiB보다 작은 볼륨을 지원하지 않습니다. 더 작은 볼륨을 요청하면 Trident는 50GiB 볼륨을 생성합니다.
- Trident는 Windows 노드에서만 실행되는 Pod에 마운트된 SMB 볼륨을 지원합니다.
- 비상업용 Azure 클라우드에 Azure NetApp Files를 배포하려면 클라우드별 Azure Resource Manager 및 인증 엔드포인트가 필요합니다. Trident와 모든 백엔드 구성이 Azure 클라우드 환경에 적합한 엔드포인트를 사용하는지 확인하십시오.

AKS에 관리형 ID 사용

Trident는 AKS 클러스터에 대한 "관리되는 ID입니다"를 지원합니다.

`tridentctl`를 사용하여 `Azure NetApp Files` 백엔드를 생성하거나 관리하는 경우 올바른 `Azure` 클라우드 환경에 대해 구성되어 있는지 확인하십시오.

관리형 ID를 사용하려면 다음이 필요합니다.

- AKS를 사용하여 구축된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 관리 ID
- `cloudProvider`이(가) `"Azure"`(으)로 설정된 상태로 Trident가 설치되었습니다

Trident 운영자

`tridentorchestrator_cr.yaml`을 편집하고 `cloudProvider`을 `"Azure"`으로 설정하세요.

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

헬름

다음 예시는 Trident를 설치하고 환경 변수 `cloudProvider`를 사용하여 `$CP`를 설정합니다:

```
helm install trident trident-operator-100.2610.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

다음 예제는 Trident를 설치하고 `cloud-provider` 플래그를 `Azure`로 설정합니다:

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

AKS에 워크로드 ID 사용

워크로드 ID를 사용하면 Kubernetes Pod가 워크로드 ID로 인증하여 Azure 리소스에 액세스할 수 있습니다.

`tridentctl`를 사용하여 `Azure NetApp Files` 백엔드를 생성하거나 관리하는 경우 올바른 `Azure` 클라우드 환경에 대해 구성되어 있는지 확인하십시오.

워크로드 ID를 사용하려면 다음이 필요합니다.

- AKS를 사용하여 구축된 Kubernetes 클러스터
- AKS Kubernetes 클러스터에 구성된 워크로드 ID 및 oidc-발급자입니다
- Trident `cloudProvider` "Azure" `cloudIdentity`워크로드 ID 값으로 설정하여 설치되었습니다.

Trident 운영자

`tridentorchestrator_cr.yaml`을 편집하고 `cloudProvider`를 `"Azure"`로 설정하세요. `cloudIdentity`를 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`로 설정하세요.

```
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
```

헬름

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 사용하여 `$CP`를 설정하고 `cloudIdentity`를 사용하여 `$CI`를 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`을 `$CP`로, `cloud-identity`을 `$CI`로 설정합니다:

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

Azure NetApp Files 백엔드를 구성할 준비를 합니다

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

지원되는 **Azure** 클라우드 환경

Trident는 다양한 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)
- Azure China

환경을 준비할 때 Azure 구독, ID 구성 및 Azure NetApp Files 리소스가 적절한 Azure 클라우드 환경에 생성되었는지 확인하십시오.

Azure NetApp Files를 처음 사용하거나 새 위치에서 사용하는 경우 Azure NetApp Files를 설정하고 NFS 볼륨을 생성하기 위해 몇 가지 초기 구성이 필요합니다. "Azure: Azure NetApp Files를 설정하고 NFS 볼륨을 생성합니다"을(를) 참조하십시오.

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

- `subscriptionID`, `tenantID`, `clientID`, `location`, 및 `clientSecret` AKS 클러스터에서 관리되는 ID를 사용하는 경우 선택 사항입니다.
- `tenantID`, `clientID`, 및 `clientSecret` AKS 클러스터에서 클라우드 ID를 사용할 때는 선택 사항입니다.
- 비상업용 Azure 클라우드에 Azure NetApp Files를 배포하려면 클라우드별 Azure Resource Manager 및 인증 엔드포인트가 필요합니다. Trident와 모든 백엔드 구성이 Azure 클라우드 환경에 적합한 엔드포인트를 사용하는지 확인하십시오.

- 용량 풀입니다. 을 참조하십시오 "Microsoft: Azure NetApp Files에 대한 용량 풀을 생성합니다".
- Azure NetApp Files에 위임된 서브넷. 을 참조하십시오 "Microsoft: Azure NetApp Files에 서브넷을 위임합니다".
- Azure NetApp Files가 활성화된 Azure 구독의 'SubscriptionID'입니다.
- `tenantID`, `clientID`, 및 `clientSecret` 에서 "앱 등록" Azure NetApp Files 서비스에 대한 충분한 권한이 있는 Azure Active Directory에서 앱 등록에서는 다음 중 하나를 사용해야 합니다.
  - 소유자 또는 참가자 역할입니다 "Azure에서 사전 정의".
  - a "사용자 지정 참가자 역할"구독 수준의 (`assignableScopes`경우) Trident에 필요한 권한으로만 제한된 다음 사용 권한이 있습니다. 사용자 지정 역할을 만든 후"Azure 포털을 사용하여 역할을 할당합니다"

    ▼ 사용자 지정 기고자 역할입니다

```
{
  "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/re
ad",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/wr
ite",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/de
lete",

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

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/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": []
      }
    ]
  }
}
```

- 적어도 하나의 "위임된 서브넷"를 포함하는 Azure `location`. `location` 매개변수는 백엔드 구성 파일의 최상위 수준에서 필요합니다. 가상 풀에 지정된 위치 값은 무시됩니다.

- 사용합니다 `Cloud Identity`을(를) 다운로드하십시오 `client ID` 에서 "사용자가 할당한 관리 ID 입니다" 에서 해당 ID를 지정합니다 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

SMB 볼륨에 대한 추가 요구사항

SMB 볼륨을 생성하려면 다음이 있어야 합니다.

- Active Directory가 구성되어 Azure NetApp Files에 연결되었습니다. 을 참조하십시오 "Microsoft: Azure NetApp Files에 대한 Active Directory 연결을 만들고 관리합니다".
- Linux 컨트롤러 노드 및 Windows Server 2022를 실행하는 Windows 작업자 노드가 있는 Kubernetes 클러스터 Trident는 Windows 노드에서만 실행되는 Pod에 마운트된 SMB 볼륨을 지원합니다.
- Azure NetApp Files가 Active Directory에 인증할 수 있도록 Active Directory 자격 증명을 포함하는 Trident 암호가 하나 이상 있어야 합니다. 비밀 생성하기 `smbcreds`:

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

- Windows 서비스로 구성된 CSI 프록시. 를 구성합니다 `csi-proxy`를 참조하십시오 "GitHub:CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우:

Azure NetApp Files 백엔드 구성 옵션 및 예

Azure NetApp Files에 대한 NFS 및 SMB 백엔드 구성 옵션에 대해 알아보고 구성 예제를 검토합니다.

백엔드 구성 옵션

Trident는 백엔드 구성(서브넷, 가상 네트워크, 서비스 수준, 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.

매개 변수	설명	기본값
'내전'	백엔드 구성 버전.	항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	"Azure-NetApp-파일"
백엔드이름	스토리지 백엔드의 사용자 지정 이름	드라이버 이름 + "_" + 임의 문자
'스크립트 ID'입니다	Azure 구독의 구독 ID입니다 AKS 클러스터에서 관리되는 ID가 설정된 경우 선택 사항입니다.	
텐antID	앱 등록에서 테넌트 ID입니다 관리 ID 또는 클라우드 ID가 AKS 클러스터에서 사용되는 경우 선택 사항입니다.	

매개 변수	설명	기본값
'클라이언트 ID'	앱 등록의 클라이언트 ID입니다 관리 ID 또는 클라우드 ID가 AKS 클러스터에서 사용되는 경우 선택 사항입니다.	
'clientSecret	앱 등록에서 클라이언트 암호 관리 ID 또는 클라우드 ID가 AKS 클러스터에서 사용되는 경우 선택 사항입니다.	
'저급'	'초표준', '프리미엄', '울트라' 중 하나	""(임의)
위치	새 볼륨을 생성할 Azure 위치의 이름입니다 AKS 클러스터에서 관리되는 ID가 설정된 경우 선택 사항입니다.	
`cloudConfiguration.cloudName`	백엔드용 Azure 클라우드 환경을 식별합니다. Azure Government를 사용하는 경우 `AzureGovernment`로 설정합니다.	
[재치 단체	검색된 자원을 필터링하기 위한 자원 그룹 목록입니다	"[]"(필터 없음)
'netap계정'	검색된 리소스를 필터링하기 위한 NetApp 계정의 목록입니다	"[]"(필터 없음)
용량풀	검색된 리소스를 필터링하기 위한 용량 풀 목록입니다	"[]"(필터 없음, 임의)
가상네트워크	위임된 서브넷이 있는 가상 네트워크의 이름입니다	""
'우방'	Microsoft.Netapp/volumes` 에 위임된 서브넷의 이름입니다	""
'네트워크 기능'	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.	""

매개 변수	설명	기본값
nfsMountOptions를 선택합니다	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`를 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 큰 경우 용량 할당에 실패합니다	""(기본적으로 적용되지 않음)
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: "\{"api":false, "method":true, "discovery":true}". 문제 해결 중이 아니며 자세한 로그 덤프가 필요한 경우가 아니면 이 방법을 사용하지 마십시오.	null입니다
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 입니다 `nfs`, `smb` 또는 null입니다. Null로 설정하면 기본적으로 NFS 볼륨이 설정됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 영역 및 영역의 목록을 나타냅니다. 자세한 내용은 을 "CSI 토폴로지를 사용합니다"참조하십시오.	
`qosType`	QoS 유형(자동 또는 수동)을 나타냅니다.	자동
`maxThroughput`	허용되는 최대 처리량을 MiB/초 단위로 설정합니다. 수동 QoS 용량 풀에서만 지원됩니다.	`4 MiB/sec`

네트워크 기능에 대한 자세한 내용은 을 참조하십시오 "Azure NetApp Files 볼륨에 대한 네트워크 기능을 구성합니다".

Azure 클라우드 환경 고려**(26.06)**

26.06 릴리스부터 Trident는 여러 Azure 클라우드 환경에서 Azure NetApp Files 백엔드를 생성하고 관리하는 것을 지원합니다.

지원되는 Azure 클라우드는 다음과 같습니다.

- Azure Commercial
- Azure Government(Azure Government/MAG)
- Azure China

Trident를 배포하거나 Azure NetApp Files 백엔드를 생성할 때 Azure Resource Manager 및 인증 엔드포인트가 Azure 클라우드 환경과 일치하는지 확인하십시오. 엔드포인트가 일치하지 않으면 `tridentctl` 인증할 수 없으며

백엔드 생성이 실패합니다.

Azure Government 백엔드의 경우 백엔드 구성에서 `cloudConfiguration.cloudName`을(를) 설정합니다:

```
cloudConfiguration:
  cloudName: AzureGovernment
```

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 구성하는 동안 가상 네트워크에 대한 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	리소스 그룹>
NetApp 계정	리소스 그룹>/<NetApp 계정>
용량 풀	리소스 그룹>/<NetApp 계정>/<용량 풀>
가상 네트워크	리소스 그룹>/<가상 네트워크>
서브넷	리소스 그룹>/<가상 네트워크>/<서브넷>

볼륨 프로비저닝

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

매개 변수	설명	기본값
엑포트 규칙	새 볼륨에 대한 엑스포트 규칙 `exportRule` CIDR 표기법을 사용하여 IPv4 주소 또는 IPv4 서브넷의 조합을 쉼표로 구분해야 합니다. SMB 볼륨에 대해 무시됩니다.	"0.0.0.0/0"
나프산디렉토리	'.snapshot' 디렉토리에 액세스합니다	`true`, `false` (명시적으로 설정).
'크기'입니다	새 볼륨의 기본 크기입니다	"100G"
유니크권한	새 볼륨의 UNIX 사용 권한(8진수 4자리) SMB 볼륨에 대해 무시됩니다.	""(미리보기 기능, 가입 시 화이트리스트 필요)

예제 설정

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 Azure NetApp Files에 위임된 모든 NetApp 계정, 용량 풀 및 서브넷을 검색하고 해당 풀과 서브넷 중 하나에 새 볼륨을 무작위로 배치합니다. `nasType`이 생략되었으므로 `nfs` 기본값이 적용되어 백엔드에서 NFS 볼륨을 프로비저닝합니다.

이 구성은 Azure NetApp Files를 시작하여 시험할 때 이상적이지만, 실제로는 프로비저닝한 볼륨에 대해 추가 범위를 제공하고 싶을 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
```

▼ **AKS**의 관리되는 **ID**입니다

이 백엔드 구성은 생략됩니다 `subscriptionID`, `tenantID`, `clientID`, 및 `clientSecret`관리되는 ID를 사용할 경우 선택 사항입니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - 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 생략됩니다 `tenantID`, `clientID`, 및 `clientSecret`클라우드 ID를 사용할 경우 선택 사항입니다.

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

▼ 용량 풀 필터를 사용한 특정 서비스 수준 구성

이 백엔드 구성은 Azure의 `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 예제

이 백엔드 구성은 Azure의 볼륨을 배치합니다. `eastus` 수동 QoS 용량 풀이 있는 위치입니다.

```
---
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 단일 서브넷에 대한 볼륨 배치 범위를 더욱 줄여주고 일부 볼륨 프로비저닝 기본값도 수정합니다.

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

다음 사항을 참조하십시오 `StorageClass` 정의는 위의 스토리지 풀을 참조합니다.

을 사용한 정의 예 **`parameter.selector`** 필드에 입력합니다

`parameter.selector`을 사용하면 각 `StorageClass`에 대해 볼륨을 호스팅하는 데 사용되는 가상 풀을 지정할 수 있습니다. 볼륨은 선택한 풀에 정의된 속성을 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 정의의 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 네임스페이스에 대한 기본 구성

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

▼ 네임스페이스별로 다른 암호 사용

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

▼ 볼륨별로 다른 암호 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.

백엔드를 생성합니다

백엔드 구성 파일을 생성한 후 다음 명령을 실행합니다.

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

상용이 아닌 Azure 클라우드를 사용하는 경우 `tridentctl`이(가) Azure 클라우드 환경에 대한 Azure Resource Manager 및 인증 엔드포인트를 사용하도록 구성되어 있는지 확인하십시오. 백엔드 생성이 실패하면 백엔드 구성을 확인하고 로그를 검토하여 원인을 파악하십시오.

```
tridentctl logs
```

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

Google Cloud NetApp 볼륨

Google Cloud NetApp Volumes 구성

Google Cloud NetApp Volumes를 Trident의 백엔드로 구성하여 Kubernetes 워크로드용 스토리지를 프로비저닝할 수 있습니다.

개요

Trident는 NAS(NFS 및 SMB)와 블록(iSCSI) 워크로드 모두에 대해 Google Cloud NetApp Volumes를 지원합니다.

- NAS 워크로드는 `google-cloud-netapp-volumes` 백엔드를 사용합니다
- 블록(iSCSI) 워크로드는 `google-cloud-netapp-volumes-san` 백엔드를 사용합니다.

NAS 볼륨은 파일 기반 스토리지를 제공하며 NFS 또는 SMB 프로토콜을 사용하여 액세스합니다. 이러한 볼륨은 여러 파드 또는 노드에서 공유 액세스를 지원합니다.

블록 볼륨은 원시 블록 스토리지를 제공하며 Kubernetes 노드에 연결된 iSCSI 장치로 액세스됩니다. 이러한 볼륨은 애플리케이션에 블록 수준 액세스가 필요한 경우에 사용됩니다.

이는 다음 환경에 적용됩니다.

- Trident 26.02 이상
- Google Kubernetes Engine(GKE) 또는 GCP에서 실행되는 Red Hat OpenShift 클러스터
- OpenShift Workload Identity 페더레이션(WIF)에는 Trident 26.10 이상이 필요합니다.
- Google Cloud NetApp Volumes 스토리지 풀

블록(iSCSI) 스토리지를 구성하려면 "블록 스토리지(iSCSI) 구성"을(를) 참조하십시오.

구성 준비

Cloud ID를 사용하면 Kubernetes 워크로드가 정적 자격 증명을 사용하는 대신 워크로드 ID로 인증하여 Google Cloud 리소스에 액세스할 수 있습니다.

워크로드 ID를 구성하기 전에 필요한 ID 리소스를 생성할 수 있는 권한이 있는 계정을 사용하십시오. GKE의 경우 Google Cloud 서비스 계정을 생성하고 구성할 수 있는 Google Cloud 권한이 필요합니다. OpenShift의 경우,

클러스터의 WIF 구성을 생성 또는 선택하고 Google Cloud 워크로드 ID 풀 및 공급자를 구성할 수 있는 권한이 필요합니다.

GKE 워크로드 ID 사용

Google Cloud NetApp Volumes에서 GKE 워크로드 ID를 사용하려면 다음이 필요합니다.

- Google Kubernetes Engine(GKE)을 사용하여 배포된 Kubernetes 클러스터
- GKE 클러스터에서 워크로드 ID가 활성화되고 노드 풀에서 메타데이터 서버가 활성화되었습니다
- Google Cloud NetApp Volumes 관리자 역할(`roles/netapp.admin`이 있는 Google Cloud 서비스 계정 또는 이에 상응하는 사용자 지정 역할
- Trident가 클라우드 공급자를 `GCP`로 설정하고 클라우드 ID 주석을 구성한 상태로 설치됨

Trident 운영자

Trident 오퍼레이터를 사용하여 Trident를 설치하려면 다음을 편집하십시오 `tridentorchestrator_cr.yaml`:

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

헬름

Helm을 사용하여 Trident를 설치할 때 클라우드 공급자와 클라우드 ID를 설정하십시오.

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl 을 선택합니다

클라우드 공급자와 클라우드 ID를 지정하여 Trident를 설치하십시오.

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
```

```
  -n trident
```

OpenShift에서 **Workload Identity Federation** 사용

Trident 26.10부터 Google Cloud Workload Identity Federation(WIF)을 사용하여 GCP에서 실행되는 OpenShift 클러스터의 Google Cloud NetApp Volumes 드라이버를 인증할 수 있습니다. 이는 자체 관리형 OpenShift Container Platform(OCP, IPI 배포 포함) 및 OpenShift Dedicated(OSD)에 적용되며, NAS (`google-cloud-netapp-volumes`) 및 SAN (`google-cloud-netapp-volumes-san`) 드라이버를 모두 지원합니다.

WIF는 프로젝션된 Kubernetes 서비스 계정 토큰을 단기 Google Cloud 자격 증명으로 교환합니다. Trident는 이러한 자격 증명을 사용하여 Google Cloud 서비스 계정을 가장하며, 해당 서비스 계정의 IAM 역할은 Google Cloud NetApp Volumes API 호출을 승인합니다. 이렇게 하면 백엔드 구성에 수명이 긴 서비스 계정 키를 저장하지 않아도 됩니다.

> OpenShift Dedicated 환경에서는 OIDC 공급자를 다른 OpenShift 구성 요소와 공유할 수 있습니다. 공급자의 `allowedAudiences` 목록이 비어 있는 경우 Google은 `https://` 접두사 유무에 관계없이 공급자의 정규 리소스 이름을 허용하므로 Trident 대상 그룹에 대해서는 공급자를 변경할 필요가 없습니다. Trident만을 위한 명시적인 대상 그룹 목록을 설정하지 마십시오. 공급자에 이미 명시적인 대상 그룹이 있는 경우 기존 값을 모두 유지하면서 Trident 대상 그룹을 추가하십시오. 목록을 교체하면 다른 OpenShift 구성 요소에서 토큰을 교환하지 못할 수 있습니다.

OpenShift 클러스터를 준비합니다**.**

OSD의 경우 클러스터를 생성할 때 GCP WIF 구성을 생성하거나 선택하십시오. 예를 들어 OpenShift Cluster Manager CLI를 사용하여 생성할 수 있습니다:

```
ocm gcp create wif-config --name <wif-name> --project <gcp-project-id>
```

클러스터 인증 유형으로 Workload Identity 페더레이션을 선택하고 WIF 구성을 클러스터와 연결합니다. 자체 관리형 OCP의 경우 클러스터의 서비스 계정 토큰 발급자를 신뢰하는 Google Cloud OIDC 공급자를 구성합니다. 클러스터 프로비저닝 및 토큰 프로젝션 단계는 OSD와 자체 관리형 OCP 간에 다릅니다. 해당 Red Hat 지침을 따르고 Trident 컨트롤러가 `/var/run/secrets/wif-token/token`에서 프로젝션된 토큰을 사용할 수 있도록 합니다.

OSD의 WIF 인증은 클러스터를 생성할 때 구성됩니다. Trident의 WIF 대상은 별도의 설치 설정입니다. Trident에 대해 WIF를 활성화하기 위해 클러스터를 다시 생성할 필요는 없습니다.

Trident를 설치하거나 구성하기 전에 설치에 사용되는 OpenShift ID에 선택한 설치 방법에 필요한 클러스터 권한이 있는지 확인하십시오. 설치 후 컨트롤러 및 노드 Pod가 정상인지 확인하십시오:

```
oc whoami
oc auth can-i create namespaces
oc auth can-i create csidrivers.storage.k8s.io
oc auth can-i create customresourcedefinitions.apiextensions.k8s.io
oc auth can-i create securitycontextconstraints.security.openshift.io

oc -n trident get pods
oc -n trident rollout status deploy/trident-controller --timeout=300s
```

```
oc -n trident rollout status ds/trident-node-linux --timeout=300s
```

WIF audience를 사용하여 **Trident** 설치

Trident는 설치 중에 대상 그룹을 구성할 때만 WIF용 토큰을 프로젝션합니다. Google Cloud 공급자와 백엔드의 `wipCredentialConfig.audience`에 동일한 대상 그룹을 사용하십시오:

```
//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>
```

선택한 설치 방법을 사용하여 `cloudProvider`을(를) `GCP`(으)로 설정하고 `gcpWIFAudience`을(를) 구성하십시오:

tridentctl 을 선택합니다

```
tridentctl install \
  --cloud-provider GCP \
  --gcp-wif-audience "<audience>" \
  --namespace trident
```

Trident 운영자

`TridentOrchestrator` 사양에 다음 필드를 설정하십시오:

```
spec:
  cloudProvider: GCP
  gcpWIFAudience: "<audience>"
```

헬름

차트를 설치할 때 다음 값을 설정하십시오:

```
helm install trident netapp-trident/trident-operator \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="<audience>"
```

OpenShift WIF에 대해 GKE `cloudIdentity` / `--cloud-identity` 옵션도 설정하지 마십시오. 설치 후 Trident 컨트롤러 배포에 패치를 적용하지 마십시오. 오퍼레이터가 수동 변경 사항을 되돌립니다. 프로젝션된 토큰은 `/var/run/secrets/wif-token/token`의 백엔드에서 사용할 수 있습니다. `trident-main` 컨테이너에는 셸이 포함되어 있지 않으므로 `oc exec`을(를) 사용하여 파일을 확인하는 대신 컨트롤러 배포의 볼륨 구성을 검사하십시오.

```
oc -n trident get deploy trident-controller -o yaml
```

Google Cloud 신뢰 및 권한 구성

클러스터에 대해 Google Cloud 워크로드 ID 풀 및 OIDC 공급자를 구성합니다. 공급자 발급자는 토큰의 `iss` 클레임과 일치해야 하며, 속성 매핑은 `assertion.sub`을(를) `google.subject`(으)로 매핑해야 합니다. 공급자의 허용 대상은 프로젝션된 토큰의 `aud` 클레임과 일치해야 합니다.

공개 `.well-known/openid-configuration` 및 JWKS 엔드포인트는 OIDC 검색을 사용할 때만 필요합니다. Google은 또한 `jwksJson` 또는 `--jwk-json-path`을(를) 사용하여 JWKS를 직접 업로드하는 것도 지원합니다.

Trident가 가장할 Google Cloud 서비스 계정에서 WIF 주체에게 `roles/iam.workloadIdentityUser` 역할을 부여합니다. 토큰 주체가 `system:serviceaccount:trident:trident-controller`인 경우, 주체의 형식은 다음과 같습니다.

```
principal://iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/subject/system:serviceaccount:trident:trident-controller
```

Google Cloud NetApp Volumes 리소스가 포함된 프로젝트에서 가장된 서비스 계정에 `roles/netapp.admin` 또는 필요한 권한이 있는 동등한 사용자 지정 역할을 부여합니다. 이 두 가지 권한 부여는 별개입니다. 첫 번째는 가장을 허용하는 권한이고, 두 번째는 스토리지 API 작업을 승인하는 권한입니다.

토큰 대상과 제공자 구성이 일치하지 않으면 Google Security Token Service에서 `invalid_grant` 대상 오류와 함께 교환을 거부할 수 있습니다. 다음을 사용하여 제공자 구성을 검토하십시오:

```
gcloud iam workload-identity-pools providers describe <provider-id> \
  --project=<wif-pool-project-id> \
  --location=global \
  --workload-identity-pool=<pool-id> \
  --format="yaml(oidc.allowedAudiences,oidc.issuerUri,attributeMapping)"
```

토큰 구성을 진단할 때 토큰의 `iss`, `aud` 및 `sub` 클레임을 공급자 및 IAM 구성과 비교하십시오. 전체 projected 토큰을 공유하거나 로그에 기록하지 마십시오.

Google Cloud NetApp Volumes 스토리지를 준비합니다**.**

백엔드에서 사용하는 리전에 스토리지 풀을 생성하거나 식별합니다. NAS 백엔드는 FILE 풀이 필요하고, SAN 백엔드는 "블록 스토리지(iSCSI) 구성"에 설명된 Flex Unified 풀을 사용합니다. 풀의 VPC 및 PSA(Private Service Access) 구성이 클러스터 네트워크에 적합한지 확인합니다.

예를 들어, 풀에 사용할 PSA 범위를 선택하기 전에 네트워크에 대한 기존 글로벌 VPC 피어링 범위를 나열합니다:

```
gcloud compute addresses list \
  --project <gcnv-project-id> \
  --global \
```

```
  --filter="purpose=VPC_PEERING AND network:<network-name>" \
  --format="table(name,address,prefixLength,status,network,purpose)"
```

`location`에 풀의 리전을 사용하고, 필요한 경우 `storagePools`에 전체 리소스 경로를 사용하여 풀을 명시적으로 선택하십시오. 리전 및 풀 위치 불일치, 잘못된 네트워크 또는 사용 불가능한 풀은 인증에 성공하더라도 검색 또는 프로비저닝을 방해할 수 있습니다.

Trident 백엔드를 구성합니다

공급자 구성 및 `wipCredentialConfig.audience` 모두에서 예상 토큰 대상 그룹을 사용하십시오. `projectNumber` 필드는 Google Cloud NetApp Volumes 프로젝트를 식별하며, 대상 그룹은 워크로드 ID 풀이 포함된 프로젝트 번호를 사용합니다. 이 예제의 자리 표시자를 환경에 맞는 값으로 바꾸십시오.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas-wif
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
    serviceLevel: Premium
  wipCredentialConfig:
    type: external_account
    audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
    subject_token_type: urn:ietf:params:oauth:token-type:jwt
    token_url: https://sts.googleapis.com/v1/token
    credential_source:
      file: /var/run/secrets/wif-token/token
    service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

SAN 백엔드의 경우 `google-cloud-netapp-volumes-san`을(를) `storageDriverName`(으)로 사용하고 해당 Flex Unified 풀 설정을 사용하십시오. 그 외의 경우 WIF 자격 증명 구성은 두 드라이버 모두 동일합니다.

백엔드를 적용하고 관찰된 상태를 확인하십시오:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

프로비저닝을 테스트하기 전에 백엔드가 `Bound` 상태이고 마지막 작업이 성공했는지 확인하십시오. 실제 `spec` 및 `status`을(를) 검사하십시오. 이전에 적용된 구성 주석은 이후의 변경 사항을 반영하지 않을 수 있습니다. `WIF`를 사용할 때는 정적 `apiKey` 자격 증명을 포함하지 마십시오.

엔드투엔드 프로비저닝 검증

백엔드를 대상으로 하는 StorageClass를 생성한 다음, 소규모 테스트 PVC를 생성합니다. 예를 들어, NAS StorageClass는 백엔드 레이블을 선택할 수 있습니다:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-wif-smoke
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  selector: "auth=wif"
allowVolumeExpansion: true
```

일치하는 `auth: wif` 레이블을 NAS 백엔드의 스토리지 항목에 추가하고, `gcnv-wif-smoke`를 사용하여 `PVC`를 생성한 다음, `Bound`에 도달하는지 확인합니다. `SAN` 스모크 테스트의 경우 `google-cloud-netapp-volumes-san`, SAN 드라이버 설정 및 적절한 Flex Unified 풀을 사용합니다. 완료되면 테스트 PVC 및 StorageClass를 삭제합니다.

일반적인 오류 해결

징후	확인해야 할 사항
`invalid_grant` 또는 대상 불일치	예상 토큰의 `aud`을(를) 제공업체의 허용된 대상 및 `wipCredentialConfig.audience`와(과) 비교하십시오.
`iam.serviceAccounts.getAccessToken` 권한이 거부되었습니다	Trident 컨트롤러의 `sub`에 대한 정확한 WIF 주체에 가장된 서비스 계정에 대한 `roles/iam.workloadIdentityUser`가 있는지 확인하십시오.
`netapp.storagePools.list` 권한이 거부되었습니다	가장된 서비스 계정에 올바른 프로젝트에 대한 `roles/netapp.admin` 등 Google Cloud NetApp Volumes 권한이 있는지 확인하십시오.
풀을 찾을 수 없거나 풀을 찾을 수 없습니다	백엔드 `location`, 풀 지역 및 유형, 네트워크, PSA 구성 및 명시적으로 구성된 스토리지 풀 리소스 경로를 확인하십시오.
백엔드는 아직 바인딩되지 않았습니다	Trident 컨트롤러 로그와 TridentBackendConfig 상태를 검사하여 근본적인 인증, 권한 또는 풀 검색 오류를 확인하십시오.

백엔드 조정 오류에 대한 컨트롤러 로그를 검사하려면:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

NAS 스토리지 구성

> Google Cloud NetApp Volumes UNIFIED 스토리지 풀의 경우, Trident는 볼륨 작업 중에 UNIFIED 전용 명명 및 검증 규칙을 적용합니다.

볼륨을 찾을 때 Trident는 여러 호환 가능한 볼륨 이름 변형(예: 하이픈 및 밑줄 형식)을 평가하여 가져오기 및 검색 신뢰성을 향상시킬 수 있습니다.

드라이버 세부 정보

Trident는 `google-cloud-netapp-volumes` Google Cloud NetApp Volumes에서 NAS 스토리지를 프로비저닝하는 드라이버를 제공합니다.

드라이버는 다음과 같은 액세스 모드를 지원합니다.

- ReadWriteOnce(RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod(RWOP)

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
`google-cloud-netapp-volumes`	NFS 를 참조하십시오 중소기업	파일 시스템	RWO, ROX, rwx, RWOP	`nfs`, `smb`

Trident NAS 백엔드 구성

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

NAS 볼륨 프로비저닝

NAS 볼륨은 `google-cloud-netapp-volumes` 백엔드를 사용하여 프로비저닝되며 NFS 및 SMB 프로토콜을 지원합니다.

NFS 볼륨용 StorageClass

NFS 볼륨을 프로비저닝하려면 `nasType`을(를) `nfs`(으)로 설정하십시오.

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

SMB 볼륨용 StorageClass

SMB 볼륨을 프로비저닝하려면 `nasType`을(를) `smb`(으)로 설정하고 자격 증명을 제공하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-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"
allowVolumeExpansion: true
```

PersistentVolumeClaim 예시 (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
```

```
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

PersistentVolumeClaim 예시(RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

NAS 볼륨은 `volumeMode: Filesystem`을 사용합니다.

SAN 워크로드를 위해 Google Cloud NetApp Volumes 구성

`google-cloud-netapp-volumes-san` 드라이버를 사용하여 Google Cloud NetApp Volumes Flex Unified 스토리지 풀에서 iSCSI 블록 볼륨을 프로비저닝하십시오.

SAN 드라이버는 블록 워크로드만 지원합니다. NAS 볼륨을 프로비저닝하려면 `google-cloud-netapp-volumes` 드라이버를 사용하십시오.

개요

드라이버는 Flex Unified 스토리지 풀에서 볼륨을 프로비저닝하고 이를 iSCSI 블록 장치로 Kubernetes 노드에 제공합니다.

드라이버는 다음 환경을 지원합니다:

- Trident 26.02 이상
- Google Kubernetes Engine(GKE) 또는 GCP에서 실행되는 Red Hat OpenShift 클러스터
- Google Cloud NetApp Volumes Flex 통합 스토리지 풀
- iSCSI 기반 워크로드

Trident 26.10부터 SAN 드라이버는 GCP에서 실행되는 자체 관리형 OpenShift Container Platform(OCP, IPI 배포 포함) 및 OpenShift Dedicated(OSD) 클러스터에서 Google Cloud Workload Identity Federation(WIF)을

지원합니다. WIF 필수 조건 및 백엔드 구성은 "OpenShift에서 Workload Identity Federation 사용"을(를) 참조하십시오. 이 페이지의 SAN 드라이버 이름과 Flex Unified 스토리지 설정을 사용하십시오.

SAN용 WIF 구성

WIF는 프로젝션된 Kubernetes 서비스 계정 토큰을 단기 Google Cloud 자격 증명으로 교환합니다. Trident는 이러한 자격 증명을 사용하여 Google Cloud NetApp Volumes API를 호출할 때 Google Cloud 서비스 계정을 가장합니다.

OpenShift 클러스터에 대한 Google Cloud 워크로드 ID 풀 및 OIDC 공급자를 구성합니다. 공급자 발급자는 토큰의 `iss` 클레임과 일치해야 하며, 속성 매핑은 `assertion.sub`을(를) `google.subject`(으)로 매핑해야 합니다.

공급자 리소스 이름을 WIF 대상으로 사용하여 Trident를 설치하십시오. 이렇게 하면 Trident가 컨트롤러 서비스 계정 토큰을 `/var/run/secrets/wif-token/token`에 투영합니다.

Trident 운영자

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: GCP
  gcpWIFAudience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

헬름

```
helm install trident <trident-operator-package> \
  --namespace trident \
  --create-namespace \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="//iam.googleapis.com/projects/<wif-pool-
project-number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

tridentctl 을 선택합니다

```
tridentctl install \
  --namespace trident \
  --cloud-provider GCP \
  --gcp-wif-audience "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

Trident 설치 및 `wipCredentialConfig.audience`에 동일한 대상을 사용하십시오. 또한 GKE `cloudIdentity` 또는 `--cloud-identity`를 구성하지 마십시오. 프로젝션된 토큰을 추가하기 위해 컨트롤러 Deployment를 패치하지 마십시오. Trident operator가 해당 변경 사항을 덮어쓸 수 있습니다.

> OpenShift Dedicated 환경에서는 OIDC 공급자를 다른 OpenShift 구성 요소와 공유할 수 있습니다. 공급자의 `allowedAudiences` 목록이 비어 있는 경우 Google은 `https://` 접두사 유무에 관계없이 공급자의 정규 리소스 이름을 허용하므로 위의 대상 그룹에 대해서는 공급자를 변경할 필요가 없습니다. Trident에 대해서만 명시적인 대상 그룹 목록을 설정하지 마십시오. 공급자에 이미 명시적인 대상 그룹이 있는 경우 기존 값을 모두 유지하면서 Trident 대상 그룹을 추가하십시오. 목록을 교체하면 다른 OpenShift 구성 요소에서 토큰을 교환하지 못할 수 있습니다.

Trident 컨트롤러 서비스 계정의 WIF 주체에게 Trident가 가장할 Google Cloud 서비스 계정에 대한 `roles/iam.workloadIdentityUser` 역할을 부여합니다. 기본 Trident 컨트롤러 서비스 계정의 토큰 주체는 `system:serviceaccount:trident:trident-controller`입니다. 설치에 맞게 구성된 주체를 사용하십시오. 가장된 서비스 계정에 Google Cloud NetApp Volumes 리소스가 포함된 프로젝트에 대한 `roles/netapp.admin` 또는 이와 동등한 권한을 부여합니다.

SAN 백엔드의 `spec`에 다음 구문을 추가하십시오:

```
wipCredentialConfig:
  type: external_account
  audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
  subject_token_type: urn:ietf:params:oauth:token-type:jwt
  token_url: https://sts.googleapis.com/v1/token
  credential_source:
    file: /var/run/secrets/wif-token/token
  service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

WIF 구성은 두 GCNV 드라이버 모두에 대해 동일합니다. SAN 백엔드는 여전히 `google-cloud-netapp-volumes-san` 및 지원되는 Flex Unified 스토리지 풀을 사용해야 합니다.

Flex Unified 스토리지 풀

Flex Unified 스토리지 풀은 iSCSI 프로토콜을 사용하여 블록 스토리지를 제공하며 SAN 프로비저닝에 필요합니다.

- Flex Unified REGIONAL 풀이 지원됩니다.
- Flex Unified ZONAL 풀은 Trident 26.02.1부터 지원됩니다.
- SAN 워크로드의 경우 **Flex** 서비스 레벨만 지원됩니다.

백엔드 `location`는 Flex Unified 풀의 지역과 일치해야 합니다. Trident가 풀을 찾지 못하면 지역, 네트워크 및 풀 리소스 경로를 확인하십시오.

Trident SAN 백엔드를 구성합니다

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

StorageClass 생성

SAN 백엔드를 구성한 후 `google-cloud-netapp-volumes-san` 드라이버를 참조하는 StorageClass를 생성합니다.

백엔드가 아닌 StorageClass에서 파일 시스템 유형을 지정하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

이 드라이버는 다음과 같은 파일 시스템 유형을 지원합니다:

- `ext4` (기본값)
- `ext3`
- `xfs`

SAN 드라이버는 Flex 서비스 레벨만 지원하며 `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions`와 같은 NAS 관련 백엔드 매개변수나 티어링 관련 설정은

사용하지 않습니다.

WIF 프로비저닝 확인

백엔드를 적용하고 TridentBackendConfig가 바인딩되었는지, 그리고 마지막 작업이 성공했는지 확인합니다:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

이 페이지의 `gcnv-san` StorageClass를 사용하여 PVC를 생성하고 `Bound`에 도달하는지 확인하십시오. 이렇게 하면 WIF 교환, 서비스 계정 가장, Google Cloud NetApp Volumes 권한 및 SAN 풀 선택이 모두 검증됩니다. 완료되면 테스트 PVC를 삭제하십시오.

백엔드 조정이 실패하면 Trident 컨트롤러 로그를 확인하십시오:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

일반적인 오류에 대해 다음을 확인하십시오:

- `invalid_grant` 대상 오류가 발생한 경우, 토큰의 `aud` 클레임을 OIDC 공급자가 허용한 대상 및 `wipCredentialConfig.audience`과(와) 비교하십시오.
- `iam.serviceAccounts.getAccessToken` 권한 오류가 발생한 경우, Trident 컨트롤러의 WIF 주체가 가장된 서비스 계정에 대해 `roles/iam.workloadIdentityUser`을(를) 가지고 있는지 확인하십시오.
- `netapp.storagePools.list` 권한 오류가 발생하는 경우, 가장된 서비스 계정에 올바른 프로젝트에 필요한 Google Cloud NetApp Volumes 권한이 있는지 확인하십시오.
- 풀을 찾을 수 없는 경우, 풀의 위치와 유형이 지원되는지, 백엔드 지역이 풀 지역과 일치하는지, 그리고 백엔드가 올바른 네트워크와 풀을 참조하는지 확인하십시오.

블록 볼륨 프로비저닝

ReadWriteOnce(RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod(RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

ROX에서 흔히 사용되는 패턴은 기존 ReadWriteOnce 볼륨을 복제하고 복제본을 읽기 전용으로 마운트하는 것입니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — 원시 블록 전용

ReadWriteMany는 `volumeMode: Block`인 경우에만 지원됩니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
```

```
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

블록 볼륨 동작

블록 볼륨은 iSCSI LUN으로 프로비저닝되고 Kubernetes 노드에 블록 디바이스로 제공됩니다.

블록 볼륨:

- iSCSI 프로토콜 사용
- 파일 시스템 및 원시 블록 프레젠테이션 지원
- Trident에 의해 연결되고 관리됩니다
- 여러 Kubernetes 액세스 모드 지원

액세스 모드

Trident에서 프로비저닝한 블록 볼륨은 다음과 같은 액세스 모드를 지원합니다.

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), 다음과 같은 경우에만 지원됩니다 `volumeMode: Block`

volumeMode 동작

`volumeMode` 필드는 블록 볼륨이 노출되는 방식을 제어합니다.

- `Filesystem` Trident가 볼륨을 포맷하고 마운트합니다.
- `Block` Trident는 장치를 연결하고 이를 원시 블록 장치로 노출합니다.

지원되는 작업

`google-cloud-netapp-volumes-san` 드라이버를 사용하여 프로비저닝된 블록 볼륨은 다음을 지원합니다:

- 생성
- 삭제
- 클론
- 스냅샷

- 크기 조정
- 가져오기

볼륨 크기 오버헤드

Google Cloud NetApp Volumes 블록 볼륨에는 내부 메타데이터 오버헤드가 포함됩니다. 따라서 커널에 보이는 장치 크기는 프로비저닝된 용량보다 작습니다.

관찰된 오버헤드:

- 볼륨 생성 시 약 300 KiB
- 크기 조정 후 최대 약 107 MiB

Google Cloud NetApp Volumes는 전체 GiB 단위로 용량 할당을 허용합니다. 사용 가능한 장치 크기가 PVC 요청을 충족하거나 초과하도록 하려면 Trident:

- 요청된 크기를 다음 정수 GiB로 올림합니다.
- 1GiB 버퍼를 추가합니다.

예를 들면 다음과 같습니다.

- PVC 요청: 100 GiB.
- 프로비저닝된 용량: 101 GiB.
- 애플리케이션에 표시되는 사용 가능한 용량: 최소 100 GiB.

Pod 예

파일 시스템 마운트 블록 볼륨**(RWO)**

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

원시 블록 디바이스(**RWX**)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

연결 및 마운트 동작

Google Cloud NetApp Volumes에서 프로비저닝된 SAN 볼륨의 경우:

- Trident는 Flex Unified 스토리지 풀에 LUN(논리 유닛 번호)을 생성합니다.
- 게시 중에 Trident는 LUN을 노드별 호스트 그룹에 매핑합니다.
- 노드 스테이징 중에 Trident는 다음을 수행합니다.
  - iSCSI 타겟에 로그인합니다
  - LUN을 검색합니다
  - 다중 경로를 구성합니다
- `volumeMode: Filesystem`Trident는 필요한 경우 장치를 포맷하고 마운트합니다.
- `volumeMode: Block`인 경우 Trident는 포맷이나 마운트 없이 장치를 연결하고 Pod에 직접 노출합니다.

> SAN 블록 볼륨은 분산 잠금 또는 쓰기 조정을 제공하지 않습니다. 여러 노드에서 블록 볼륨에 액세스할 때(ReadWriteMany 사용 시 `volumeMode: Block`) 애플리케이션 또는 파일 시스템이 동시성을 관리해야 합니다.

Google Cloud NetApp 볼륨 백엔드 구성을 준비합니다

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

NFS 또는 **SMB** 볼륨의 사전 요구 사항

Google Cloud NetApp Volumes를 처음 사용하거나 새 위치에서 사용하는 경우 Google Cloud NetApp Volumes를

설정하고 NFS 또는 SMB 볼륨을 생성하려면 몇 가지 초기 구성이 필요합니다. "시작하기 전에"을 참조하십시오.

Google Cloud NetApp 볼륨 백엔드를 구성하기 전에 다음 사항이 있는지 확인하십시오.

- Google Cloud NetApp Volumes 서비스로 구성된 Google Cloud 계정 을 "Google Cloud NetApp 볼륨 "참조하십시오.
- Google Cloud 계정의 프로젝트 번호입니다. 을 "프로젝트 식별"참조하십시오.
- NetApp 볼륨 관리자) 역할을 가진 Google Cloud 서비스 계정입니다. (`roles/netapp.admin` 을 "ID 및 액세스 관리 역할 및 권한"참조하십시오.
- GCNV 계정에 대한 API 키 파일입니다. 을 참조하십시오 "서비스 계정 키를 생성합니다"
- 스토리지 풀입니다. 을 "스토리지 풀 개요"참조하십시오.

Google Cloud NetApp 볼륨에 대한 액세스를 설정하는 방법에 대한 자세한 내용은 를 참조하십시오 "Google Cloud NetApp 볼륨에 대한 액세스 설정".

Google Cloud NetApp 볼륨 백엔드 구성 옵션 및 예

Google Cloud NetApp 볼륨의 백엔드 구성 옵션에 대해 알아보고 구성 예제를 검토합니다.

백엔드 구성 옵션

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

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	의 값을 `storageDriverName` "google-cloud-netapp-volumes"로 지정해야 합니다.
백엔드이름	(선택 사항) 스토리지 백엔드의 사용자 지정 이름입니다	드라이버 이름 + "_" + API 키의 일부
`storagePools`	볼륨 생성을 위한 스토리지 풀을 지정하는 데 사용되는 선택적 매개 변수입니다.	
'프로젝트 번호'	Google Cloud 계정 프로젝트 번호입니다. 이 값은 Google Cloud 포털 홈 페이지에서 확인할 수 있습니다.	
위치	Trident가 GCNV 볼륨을 생성하는 Google Cloud 위치입니다. 교차 지역 Kubernetes 클러스터를 생성할 경우, 에서 생성된 볼륨을 `location` 여러 Google Cloud 지역의 노드에 예약된 워크로드에 사용할 수 있습니다. 지역 간 트래픽에는 추가 비용이 발생합니다.	

매개 변수	설명	기본값
아피키	역할이 지정된 Google Cloud 서비스 계정의 API 키입니다. `netapp.admin` 여기에는 Google Cloud 서비스 계정의 개인 키 파일(백엔드 구성 파일에 verbatim 복사)의 JSON 형식 콘텐츠가 포함됩니다. `apiKey`, , , , , 키를 사용하려면 키-값 쌍을 포함해야 합니다 `type` `project_id` `client_email` `client_id` `auth_uri`. `token_uri` `auth_provider_x509_cert_url`, , 및 `client_x509_cert_url`.	
nfsMountOptions를 선택합니다	NFS 마운트 옵션에 대한 세밀한 제어	"nfsvers=3"
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다.	""(기본적으로 적용되지 않음)
'저급'	스토리지 풀 및 해당 볼륨의 서비스 레벨입니다. 값은 `flex`, `standard`, `premium`또는 `extreme`입니다.	
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""
네트워크	GCNV 볼륨에 사용되는 Google Cloud 네트워크입니다.	
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: `{"api":false, "method":true}` 문제 해결 중이 아니며 자세한 로그 덤프가 필요한 경우가 아니면 이 방법을 사용하지 마십시오.	null입니다
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 입니다 `nfs`, `smb` 또는 null입니다. Null로 설정하면 기본적으로 NFS 볼륨이 설정됩니다.	`nfs`
`supportedTopologies`	이 백엔드에서 지원하는 영역 및 영역의 목록을 나타냅니다. 자세한 내용은 을 "CSI 토폴로지를 사용합니다"참조하십시오. 예를 들면 다음과 같습니다. `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

볼륨 프로비저닝 옵션

에서 기본 볼륨 프로비저닝을 제어할 수 있습니다 `defaults` 구성 파일의 섹션입니다.

매개 변수	설명	기본값
엑포트 규칙	새 볼륨의 내보내기 규칙. IPv4 주소 조합을 쉼표로 구분하여 나열해야 합니다.	"0.0.0.0/0"
나프산디렉토리	'.snapshot' 디렉토리에 액세스합니다	`true`, `false` (기본 동작은 다를 수 있습니다. 명시적으로 설정) NFSv3의 경우 "false"

매개 변수	설명	기본값
안산예비역	스냅숏용으로 예약된 볼륨의 백분율입니다	""(기본값 0 적용)
유니크권한	새 볼륨의 UNIX 사용 권한(8진수 4자리)	""

예제 설정

다음 예에서는 대부분의 매개 변수를 기본값으로 두는 기본 구성을 보여 줍니다. 이는 백엔드를 정의하는 가장 쉬운 방법입니다.

▼ 최소 구성

이는 절대적인 최소 백엔드 구성입니다. 이 구성을 통해 Trident은 Google Cloud NetApp 볼륨에 위임된 모든 스토리지 풀을 구성된 위치에서 검색하고 해당 풀 중 하나에 무작위로 새 볼륨을 배치합니다. 이 생략되므로 `nasType` `nfs` 기본값이 적용되고 백엔드에서 NFS 볼륨에 대한 프로비저닝이 수행됩니다.

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



`<id_value>` 및 `<key_value>`를 서비스 계정 자격 증명으로 바꾸십시오.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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** 볼륨을 위한 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 사용하여 구성합니다

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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 섹션에 정의되어 `storage` 있습니다. 서로

다른 서비스 수준을 지원하는 여러 스토리지 풀이 있고 Kubernetes에서 이러한 풀을 나타내는 스토리지 클래스를 생성하려는 경우에 유용합니다. 가상 풀 레이블은 풀을 구분하는 데 사용됩니다. 예를 들어, 아래 예에서는 `performance` 가상 풀을 구분하는 데 레이블 및 `serviceLevel` 유형이 사용됩니다.

일부 기본값을 모든 가상 풀에 적용할 수 있도록 설정하고 개별 가상 풀에 대한 기본값을 덮어쓸 수도 있습니다. 다음 예에서는 `snapshotReserve` 모든 가상 풀에 대해 기본값으로 사용됩니다. `exportRule`

자세한 내용은 을 "가상 풀"참조하십시오.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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**

GKE 클라우드 ID를 사용하여 Trident를 설치하면 백엔드는 워크로드 ID로 인증하고 `apiKey` 블록을 생략합니다. 클라우드 ID는 Google Cloud NetApp Volumes 드라이버 모두에 대해 동일한 방식으로 구성됩니다. 유일한 차이점은 `storageDriverName` 값입니다. NAS(NFS 또는 SMB) 볼륨에는 `google-cloud-netapp-volumes`을 사용하고 SAN(iSCSI 블록) 볼륨에는 `google-cloud-netapp-volumes-san`을 사용합니다.

> ⓘ `google-cloud-netapp-volumes-san` 드라이버에 대한 클라우드 ID 지원을 위해서는 Trident 26.06 이상이 필요합니다. SAN 볼륨은 Flex Unified 스토리지 풀에서 프로비저닝되므로 `serviceLevel`을(를) `flex`(으)로 설정하십시오.

NAS 드라이버(<code>google-cloud-netapp-volumes</code>)

```
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
```

SAN 드라이버(<code>google-cloud-netapp-volumes-san</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

▼ 지원되는 토폴로지 구성

Trident은 지역 및 가용 영역을 기준으로 워크로드에 대한 볼륨을 손쉽게 프로비저닝할 수 있도록 지원합니다. `supportedTopologies`이 백엔드 구성의 블록은 백엔드당 영역 및 영역 목록을 제공하는 데 사용됩니다. 여기에 지정한 지역 및 영역 값은 각 Kubernetes 클러스터 노드의 레이블에 있는 지역 및 영역 값과 일치해야 합니다. 이러한 영역 및 영역은 스토리지 클래스에서 제공할 수 있는 허용 가능한 값 목록을 나타냅니다. 백엔드에서 제공되는 영역 및 영역의 하위 집합이 포함된 스토리지 클래스의 경우 Trident는 언급한 영역 및 영역에 볼륨을 생성합니다. 자세한 내용은 을 "CSI 토폴로지를 사용합니다"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 실행합니다.

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

백엔드가 성공적으로 생성되었는지 확인하려면 다음 명령을 실행합니다.

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

```
tridentctl logs
```

구성 파일의 문제를 확인하고 수정한 후 백엔드를 삭제하고 create 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"
```

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

를 사용하면 `parameter.selector` 볼륨을 호스팅하는 데 사용되는 에 대해 을 지정할 수 `StorageClass` "가상 풀입니다" 있습니다. 볼륨은 선택한 풀에 정의된 측면을 갖습니다.

```
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 을 "스토리지 클래스를 생성합니다"참조하십시오.

SMB 볼륨에 대한 정의의 예

`node-stage-secret-name`, 및 를 사용하여 `nasType` `node-stage-secret-namespace` SMB 볼륨을 지정하고 필요한 Active Directory 자격 증명을 제공할 수 있습니다. 사용 권한이 있거나 없는 모든 Active Directory 사용자/암호는 노드 단계 비밀에 사용할 수 있습니다.

▼ 기본 네임스페이스에 대한 기본 구성

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

▼ 네임스페이스별로 다른 암호 사용

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
```

```
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

▼ 볼륨별로 다른 암호 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 바인딩되어 있는지 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
```

Google Cloud NetApp Volumes에 대한 자동 계층화 구성

자동 계층화는 볼륨 프로비저닝 중에 Trident 백엔드 매개변수와 PersistentVolumeClaim 주석을 통해 구성됩니다. Trident를 사용하여 Google Cloud NetApp Volumes에 대한 자동 계층화를 구성할 수 있습니다.

개요

자동 계층화 기능을 통해 Trident는 비활성 데이터를 성능 계층에서 용량 계층으로 자동으로 이동시키는 볼륨을 프로비저닝할 수 있습니다. 이를 통해 스토리지 비용을 절감하는 동시에 자주 액세스하는 데이터의 성능을 유지할 수 있습니다.

Trident는 볼륨 생성 시에만 자동 계층화 설정을 적용합니다. Trident 26.10에서는 프로비저닝 후 변경이 지원되지 않습니다.

개념

자동 계층화

자동 계층화는 접근 패턴에 따라 접근 빈도가 낮은 데이터를 성능 계층에서 용량 계층으로 이동시킵니다. 데이터 이동은 비동기적으로 발생하며 즉시 이루어지지 않습니다.

계층화 정책

계층화 정책은 볼륨에 대해 자동 계층화를 활성화할지 여부를 결정합니다.

다음 정책이 지원됩니다. * `auto`: 액세스 패턴에 따른 자동 계층화 활성화 * `none`: 자동 계층화 비활성화

냉방 일수

냉각일은 데이터 블록이 계층화 대상이 되기 전에 비활성 상태로 유지되어야 하는 최소 일수를 지정합니다. 냉각일은 계층화 정책이 `auto`로 설정된 경우에만 적용됩니다.

구성 모델

구성 범위

자동 계층화는 여러 범위에서 구성할 수 있습니다.

- 스토리지 풀 범위 풀에서 프로비저닝된 모든 볼륨에 적용됩니다.
- 볼륨 범위 PersistentVolumeClaim 주석을 통해 단일 볼륨에 적용됩니다.

Trident는 각 설정이 정의된 위치를 기반으로 유효한 구성을 결정합니다.

구성 우선 순위

동일한 설정이 여러 범위에서 정의된 경우 Trident는 다음 우선순위를 적용합니다.

1. PersistentVolumeClaim 주석
2. Trident 백엔드 구성

3. 스토리지 풀 기본값

우선 순위가 높은 설정이 하위 수준 값을 재정의합니다.

Trident 26.10에서 지원되는 기능

Trident 26.10는 Google Cloud NetApp Volumes에 대해 다음과 같은 자동 계층화 기능을 지원합니다.

- 볼륨 프로비저닝 중 자동 계층화 설정 또는 해제
- Trident 백엔드 구성에서 계층화 정책 정의
- PVC 주석을 사용하여 볼륨별 계층화 정책 및 냉각 일수 재정의
- 자동 계층화가 활성화된 볼륨에 대한 냉각 일수 구성

Trident 26.10에서 지원되지 않는 기능

다음 작업은 지원되지 않습니다.

- 볼륨 생성 후 자동 계층화 설정 수정
- Kubernetes 업데이트를 사용하여 기존 볼륨의 계층화 정책 변경
- Trident에서 관리하는 프로비저닝 워크플로 외부에서 자동 계층화 설정 적용

백엔드 구성 매개 변수

다음 매개변수는 Trident 백엔드 구성에 정의될 때 자동 계층화 동작을 제어합니다.

매개 변수	필수 요소입니다	설명
tieringPolicy	아니요	볼륨에 대한 계층화 정책 (`auto` 또는 `none`)
tieringMinimumCoolingDays	아니요	데이터가 계층화되기 전의 비활성 일수(범위: 2-183, 기본값: 31)

PersistentVolumeClaim 어노테이션을 사용한 볼륨 수준 재정의

지원되는 주석

PersistentVolumeClaim 어노테이션을 사용하면 볼륨별로 자동 계층화 설정을 재정의할 수 있습니다.

주석	설명
`trident.netapp.io/tieringPolicy`	볼륨에 대한 티어링 정책을 재정의합니다
`trident.netapp.io/tieringMinimumCoolingDays`	볼륨에 대한 쿨링 일수 값을 재정의합니다

예시: 자동 계층화 재정의가 적용된 **PersistentVolumeClaim**

```
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

동작 및 제한 사항

프로비저닝 동작

- 자동 계층화 설정은 볼륨 생성 시에만 평가 및 적용됩니다.
- Trident는 프로비저닝 후 계층화 구성을 조정하지 않습니다.
- 계층화 정책이 `none`로 설정된 경우 쿨링 기간은 무시됩니다.

플랫폼 제한 사항

- 자동 계층화는 NAS 볼륨(NFS 및 SMB)에서만 지원됩니다.
- 블록 볼륨(iSCSI)은 자동 계층화를 지원하지 않습니다.
- Google Cloud NetApp Volumes 스토리지 풀은 Google Cloud에서 자동 계층화가 활성화되어 있어야 합니다.

지원되는 값

- `tieringMinimumCoolingDays`의 유효 범위: 2~183
- 기본값: 31

Google Cloud NetApp Volumes ONTAP 모드 스토리지 풀 구성

Trident 26.06 버전부터 기존 `ontap-san` 및 `ontap-nas` 백엔드를 구성하여 Google Cloud NetApp Volumes(GCNV) ONTAP 모드 스토리지 풀에 스토리지를 프로비저닝할 수 있습니다. 이 기능은 기술 미리 보기로 제공됩니다.

참고	이 기능은 Trident 26.10의 기술 미리보기 기능입니다. 기술 미리보기 기능은 실제 운영 환경에서 사용할 수 없습니다. 기능, 구성 필드 및 지원 매트릭스는 정식 출시(GA) 전에 변경될 수 있습니다.

작동 원리

Trident는 기존 `ontap-san` 및 `ontap-nas` 드라이버 로직을 재사용하고 ONTAP REST 호출을 GCNV 프록시 엔드포인트를 통해 라우팅합니다. 이 방식을 통해 GCNV에서 호스팅되는 ONTAP 클러스터의 스토리지를 프로비저닝하고 관리하는 동시에 직접 ONTAP 백엔드에 사용하는 것과 동일한 운영 워크플로를 유지할 수 있습니다.

ONTAP 모드는 새로운 스토리지 드라이버를 도입하지 않습니다. `gcnv` 구성 블록을 `ontap-san` 또는 `ontap-nas` 백엔드에 추가하여 백엔드별로 활성화할 수 있습니다.

프로비저닝 경로는 다음과 같습니다.

`PersistentVolumeClaim` → Trident (`ontap-san` or `ontap-nas`) → ONTAP REST 클라이언트 → GCNV 프록시 → GCNV 풀의 ONTAP 클러스터

지원되는 범위 및 지원되지 않는 범위

이 기술 미리보기 버전은 다음을 지원합니다.

- 드라이버: `ontap-san`(iSCSI) 및 `ontap-nas`(NFS 또는 SMB).
- 프록시를 통한 라이프사이클 작업: 백엔드 초기화, 볼륨 생성 및 삭제, 관련 ONTAP REST 흐름.
- 인증 모델: 워크로드 ID 풀(WIP), 서비스 계정 키 및 애플리케이션 기본 자격 증명(ADC) 대체 방식.

다음 사항은 이번 기술 미리보기의 범위에서 제외됩니다.

- `ontap-san-economy`, `ontap-nas-economy` 및 `ontap-nas-flexgroup` 드라이버.
- ASA r2 퍼스낼리티 경로.
- ZAPI 대체 기능입니다. ONTAP 모드는 ONTAP REST만 사용합니다.

필수 구성 요소

ONTAP 모드 백엔드를 구성하기 전에 다음 사항을 확인하십시오.

- Trident 26.06 이상 버전.
- 대상 프로젝트 및 위치에 GCNV ONTAP 모드 스토리지 풀이 있어야 합니다.
- 사용자 환경에 맞게 범위가 지정된 프록시 작업에 대한 GCP IAM 권한입니다.
- `proxyURL`, `projectNumber`, `location` 및 `poolID`를 포함하는 백엔드 구성입니다.
- iSCSI SAN 워크플로우의 경우, 플랫폼에 필요한 노드 측 iSCSI 및 다중 경로 필수 조건을 확인하십시오. "작업자 노드를 준비합니다"을 참조하십시오.

백엔드 구성

A `TridentBackendConfig`는 `Kubernetes Secret`을 참조하기 위해 `credentials`를 필요로 합니다. `ONTAP` 모드의 경우, `Secret`은 `ONTAP` `managementLIF` / `username` / `password`가 아닌 GCP 프록시 액세스에 대한 자격 증명을 제공합니다. 인증 방법에 따라 서비스 계정 키 필드를 포함하거나 비워둘 수 있습니다. `svm`는 선택 사항입니다. 현재 각 GCNV ONTAP 모드 풀에는 하나의 SVM이 있으며, 생략할 경우 Trident는 풀에서 이를 파생합니다.

서비스 계정 키 인증의 경우 네이티브 GCNV와 동일한 패턴을 사용하십시오. 비민감 필드는 `gcnv.apiKey`에, `private_key` 및 `private_key_id`는 시크릿에 포함합니다. "Google Cloud NetApp Volumes 구성 예시"을(를) 참조하십시오. 워크로드 ID 풀 또는 GKE 클라우드 ID 인증의 경우 인증의 예제를 참조하십시오.

ontap-san 백엔드

```
apiVersion: trident.netapp.io/v1
```

```
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
  svm: "<svm-name>"
```

ontap-nas 백엔드

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      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: "<client-x509-cert-url>"
  svm: "<svm-name>"
```

비밀

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <private-key>
    -----END PRIVATE KEY-----
```

참고

`TridentBackendConfig` 사양에 `private_key` 또는 `private_key_id`을 넣지 마십시오. CRD에서 이를 거부합니다. 블록 스토리지에는 동일한 `gcnv` 및 자격 증명 패턴을 가진 `ontap-san`을 사용하십시오.

인증

Trident는 다음 순서로 프록시 액세스에 대한 자격 증명을 확인합니다:

1. 워크로드 ID 풀(WIP)
2. 서비스 계정 키
3. 애플리케이션 기본 자격 증명(ADC)

이번 Tech Preview에서는 다음 사항을 준수하십시오:

- 가능한 경우 Workload Identity Pool을 우선적으로 사용하십시오.
- 버전 관리되는 백엔드 파일에 개인 키를 직접 포함시키지 마십시오. 자격 증명은 Kubernetes 시크릿에 저장하십시오.
- 서비스 계정에 최소 권한 IAM 스코프를 적용합니다.

`gcnv` 블록의 `location`은(는) 풀의 위치와 일치해야 합니다. 영역 풀에는 영역을, 지역 풀에는 지역을 사용하십시오.

워크로드 **ID** 풀

`gcnv.wipCredential`은(는) `TridentBackendConfig`에 구성된 명시적 외부 계정 페더레이션에 사용하십시오. OpenShift에서 Trident는 `/var/run/secrets/wif-token/token`에 토큰을 프로젝션합니다. 이는 Trident가 OpenShift WIF에 대해 프로젝션하는 유일한 토큰 경로이므로 `credential_source.file`에 사용하십시오. xref:{relative_path}gcnv.html#openshift-wif["워크로드 ID 페더레이션을 사용하여 Google Cloud NetApp Volumes 구성"]에 설명된 대로 WIF 대상으로 Trident를 설치하십시오. `credentials` 필드는 여전히 Secret을 참조하지만, Secret에는 `private_key`이(가) 포함되어서는 안 됩니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wip
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wip
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    wipCredential:
      type: external_account
      audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<wif-pool-
id>/providers/<provider-id>"
      subject_token_type: "urn:ietf:params:oauth:token-type:jwt"
      token_url: "https://sts.googleapis.com/v1/token"
```

```
      service_account_impersonation_url:
"https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken"
      credential_source:
        file: "/var/run/secrets/wif-token/token"
```

GKE 클라우드 ID

GKE의 경우, "GKE 워크로드 ID 사용"에 설명된 대로 설치 시 Trident의 클라우드 ID를 구성하십시오. 백엔드는 애플리케이션 기본 자격 증명(ADC)을 사용하므로, `gcnv.apiKey` 및 `gcnv.wipCredential`를 모두 생략하십시오. `credentials` 필드는 여전히 Secret을 참조하지만, 그 안에 `private_key`를 넣지 마십시오.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wi
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
```

유효성 검사 및 초기화 동작

Trident는 초기화 중에 ONTAP 모드 백엔드를 검증하고 부분 초기화 대신 즉시 실패합니다.

필수 `gcnv` 필드가 누락되었거나, 드라이버가 ONTAP 모드에서 지원되지 않거나, 자격 증명 확인에 실패하면 백엔드 초기화가 실패합니다.

알려진 제한 사항

- 이 기능은 기술 미리보기 버전입니다. 정식 출시 전에 동작 방식 및 지원되는 매트릭스가 변경될 수 있습니다.
- ONTAP 모드에서는 ZAPI 폴백이 사용되지 않습니다.
- 프로토콜 및 노드 준비 요구 사항은 여전히 적용됩니다. 예를 들어, iSCSI 워크플로에는 "작업자 노드를 준비합니다"에 설명된 노드 측 필수 조건이 필요합니다.
- 기존 ONTAP 및 GCNV 환경 제약 조건은 여전히 적용됩니다.

업그레이드 및 호환성

ONTAP 모드는 `gcnv` 구성 블록을 통해 백엔드별로 선택적으로 활성화할 수 있습니다:

- 기존의 GCNV 이외의 ONTAP 백엔드는 영향을 받지 않습니다.
- 직접 ONTAP 백엔드와 ONTAP 모드 프록시 백엔드를 결합한 혼합 배포는 백엔드 수준 구성을 통해 지원됩니다. 각 백엔드는 독립적으로 구성됩니다.
- ONTAP 모드 사용을 중지하려면 해당 백엔드를 제거하거나 교체하십시오. 전역 스위치는 필요하지 않습니다.

다음 단계

TBC 전에 시크릿을 적용하십시오. 예시와 일치하도록 `kubectl apply` 및 `-n trident`를 사용하십시오.

NetApp HCI 또는 SolidFire 백엔드를 구성합니다

Trident 설치에서 Element 백엔드를 생성하고 사용하는 방법에 대해 알아봅니다.

요소 드라이버 세부 정보

Trident는 `solidfire-san` 클러스터와 통신할 수 있는 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(rwx), *ReadWriteOncePod*(RWOP)입니다.

`solidfire-san`스토리지 드라이버는 _FILE_AND_BLOCK_VOLUME 모드를 지원합니다. 볼륨 모드의 경우 `Filesystem` Trident는 볼륨을 생성하고 파일 시스템을 생성합니다. 파일 시스템 유형은 StorageClass에 의해 지정됩니다.

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'솔더불-산'	iSCSI	블록	RWO, ROX, rwx, RWOP	파일 시스템이 없습니다. 원시 블록 장치.
'솔더불-산'	iSCSI	파일 시스템	RWO, 공화당	xfs, ext3, ext4

시작하기 전에

Element 백엔드를 생성하기 전에 다음이 필요합니다.

- Element 소프트웨어를 실행하는 지원되는 스토리지 시스템

- 볼륨을 관리할 수 있는 NetApp HCI/SolidFire 클러스터 관리자 또는 테넌트 사용자에 대한 자격 증명
- 모든 Kubernetes 작업자 노드에 적절한 iSCSI 툴이 설치되어 있어야 합니다. 을 참조하십시오 "작업자 노드 준비 정보".

백엔드 구성 옵션

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	항상 "SolidFire-SAN"
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	"SolidFire_" + 스토리지(iSCSI) IP 주소입니다
끝점	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP입니다	
'VIP'	스토리지(iSCSI) IP 주소 및 포트	
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다.	""
테난트이름	사용할 테넌트 이름(찾을 수 없는 경우 생성됨)	
이니토IFace	iSCSI 트래픽을 특정 호스트 인터페이스로 제한합니다	"기본값"
'UseCHAP'입니다	CHAP를 사용하여 iSCSI를 인증합니다. Trident는 CHAP를 사용합니다.	참
"액세스 그룹"	사용할 액세스 그룹 ID 목록입니다	"Trident"라는 액세스 그룹의 ID를 찾습니다.
'유형'	QoS 사양	
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다	""(기본적으로 적용되지 않음)
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true}	null입니다

경고	문제 해결 및 자세한 로그 덤프가 필요한 경우가 아니면 debugTraceFlags를 사용하지 마십시오.

예 **1:** 에 대한 백엔드 구성 **`solidfire-san`** 세 가지 볼륨 유형을 가진 드라이버

이 예에서는 CHAP 인증을 사용하는 백엔드 파일을 보여 주고 특정 QoS 보장을 포함하는 세 가지 볼륨 유형을 모델링합니다. 그런 다음 "IOPS" 스토리지 클래스 매개 변수를 사용하여 각 스토리지 클래스를 사용할 스토리지 클래스를 정의할 가능성이 높습니다.

```
---
```

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

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

이 예에서는 가상 풀과 이를 다시 참조하는 StorageClasses와 함께 구성된 백엔드 정의 파일을 보여 줍니다.

Trident는 용량 할당 시 스토리지 풀에 있는 레이블을 백엔드 스토리지 LUN에 복제합니다. 편의를 위해 스토리지 관리자는 가상 풀 및 그룹 볼륨별로 레이블을 레이블별로 정의할 수 있습니다.

아래 표시된 샘플 백엔드 정의 파일에서 특정 기본값은 를 설정하는 모든 스토리지 풀에 대해 설정됩니다 `type` 실버. 가상 풀은 에 정의되어 있습니다 `storage` 섹션을 참조하십시오. 이 예에서는 일부 스토리지 풀이 자체 유형을 설정하고 일부 풀은 위에 설정된 기본값을 재정의합니다.

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

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

다음 StorageClass 정의는 위의 가상 풀을 참조합니다. 를 사용합니다 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다. 선택한 가상 풀에 볼륨이 정의되어 있습니다.

첫 번째 StorageClass(`solidfire-gold-four`)가 첫 번째 가상 풀에 매핑됩니다. 이 수영장은 금색 연주를 제공하는 유일한 수영장입니다. `Volume Type QoS` Last StorageClass(`solidfire-silver`)는 은색 성능을

제공하는 모든 스토리지 풀을 호출합니다. Trident는 어떤 가상 풀이 선택되었는지 결정하고 스토리지 요구 사항이 충족되는지 확인합니다.

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

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

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

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

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

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

자세한 내용을 확인하십시오

- "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 정보입니다

Trident는 ONTAP 클러스터와 통신할 수 있도록 다음과 같은 SAN 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(rwx), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'ONTAP-SAN'	FC를 통한 iSCSI SCSI	블록	RWO, ROX, rwx, RWOP	파일 시스템이 없습니다. 원시 블록 디바이스입니다
'ONTAP-SAN'	FC를 통한 iSCSI SCSI	파일 시스템	RWO, 공화당 파일 시스템 볼륨 모드에서는 ROX 및 rwx를 사용할 수 없습니다.	xfs, ext3, ext4
'ONTAP-SAN'	NVMe/TCP 을 참조하십시오 NVMe/TCP 에 대한 추가 고려사항.	블록	RWO, ROX, rwx, RWOP	파일 시스템이 없습니다. 원시 블록 디바이스입니다
'ONTAP-SAN'	NVMe/TCP 을 참조하십시오 NVMe/TCP 에 대한 추가 고려사항.	파일 시스템	RWO, 공화당 파일 시스템 볼륨 모드에서는 ROX 및 rwx를 사용할 수 없습니다.	xfs, ext3, ext4
ONTAP-SAN-이코노미	iSCSI	블록	RWO, ROX, rwx, RWOP	파일 시스템이 없습니다. 원시 블록 디바이스입니다

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
ONTAP-SAN-이코노미	iSCSI	파일 시스템	RWO, 공화당 파일 시스템 볼륨 모드에서는 ROX 및 rwx를 사용할 수 없습니다.	xfs, ext3, ext4

경고

- 사용 `ontap-san-economy` 영구 볼륨 사용 수가 보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한".
- 사용 `ontap-nas-economy` 영구 볼륨 사용 수가 보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 및 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마십시오 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우
- ONTAP 볼륨 자동 크기 조정은 `ontap-san` 및 `ontap-san-economy`에서 지원됩니다. ONTAP NAS 드라이버에서는 지원되지 않습니다. 자세한 내용 및 제한 사항은 "ONTAP 볼륨 자동 크기 조정을 구성합니다"을 참조하십시오.

사용자 권한

Trident는 ONTAP 또는 SVM 관리자로 실행해야 하며, 일반적으로 클러스터 사용자 `vsadmin` 또는 SVM 사용자 또는 같은 역할을 가진 다른 이름의 사용자를 사용할 `admin` 것입니다. Amazon FSx for NetApp ONTAP 배포의 경우 Trident은 클러스터 사용자 또는 `vsadmin` SVM 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행하거나 `fsxadmin` 동일한 역할을 가진 다른 이름의 사용자를 실행해야 합니다. `fsxadmin`사용자는 클러스터 관리자를 제한적으로 대체합니다.

참고

`limitAggregateUsage`매개 변수를 사용하려면 클러스터 관리 권한이 필요합니다. Trident와 함께 `Amazon FSx for NetApp ONTAP`를 사용할 때 `limitAggregateUsage` 매개 변수는 및 `fsxadmin` 사용자 계정에서 작동하지 `vsadmin` 않습니다. 이 매개 변수를 지정하면 구성 작업이 실패합니다.

Trident 드라이버가 사용할 수 있는 더 제한적인 역할을 ONTAP 내에 만들 수 있지만 권장하지 않습니다. Trident의 대부분의 새로운 릴리즈에서는 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

NVMe/TCP에 대한 추가 고려사항

Trident는 다음과 같은 드라이버를 사용하여 비휘발성 메모리 익스프레스(NVMe) 프로토콜을 `ontap-san` 지원합니다.

- IPv6
- NVMe 볼륨의 스냅샷 및 클론
- NVMe 볼륨 크기 조정
- Trident에서 라이프사이클을 관리할 수 있도록 Trident 외부에서 생성된 NVMe 볼륨을 가져옵니다
- NVMe 네이티브 다중 경로
- K8 노드의 정상 또는 비정상적으로 종료 (24.06)

Trident는 다음을 지원하지 않습니다.

- NVMe에서 기본적으로 지원되는 DH-HMAC-CHAP
- DM(Device Mapper) 경로 다중화
- Linux Unified Key Setup (LUKS) 암호화

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

ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 합니다

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

요구 사항

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

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

또한 둘 이상의 드라이버를 실행하고 둘 중 하나를 가리키는 스토리지 클래스를 생성할 수도 있습니다. 예를 들어, ONTAP-SAN 드라이버와 ONTAP-SAN-이코노미 클래스를 사용하는 '기본 클래스'를 사용하는 'san-dev' 클래스를 구성할 수 있습니다.

모든 Kubernetes 작업자 노드에는 적절한 iSCSI 툴이 설치되어 있어야 합니다. 을 참조하십시오 "작업자 노드를 준비합니다" 를 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대한 보장하기 위해 admin 또는 vsadmin과 같은 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 및 사용할 경우 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값이 있어야 합니다(권장).

자격 증명 기반 방법과 인증서 기반 방법 간에 이동하기 위해 기존 백엔드를 업데이트할 수 있습니다. 그러나 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

경고	자격 증명과 인증서 * 를 모두 제공하려고 하면 구성 파일에 둘 이상의 인증 방법이 제공된다는 오류가 발생하여 백엔드 생성이 실패합니다.

자격 증명 기반 인증을 사용합니다

Trident은 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자에 대한 자격 증명이 필요합니다. 또는 `vsadmin` 과 같은 미리 정의된 표준 역할을 사용하는 것이 좋습니다 `admin`. 따라서 향후 Trident 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리즈와의 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 만들어 Trident와 함께 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

YAML

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

JSON을 참조하십시오

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

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 위치라는 점에 유의하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 암호로 저장됩니다. 백엔드의 생성 또는 업데이트는 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes/스토리지 관리자가 수행할 수 있는 관리 전용 작업입니다.

인증서 기반 인증 활성화

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

- clientCertificate: Base64로 인코딩된 클라이언트 인증서 값입니다.
- clientPrivateKey: Base64 - 연결된 개인 키의 인코딩된 값입니다.
- TrustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

```
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. 신뢰할 수 있는 CA 인증서를 ONTAP 클러스터에 추가합니다. 이는 스토리지 관리자가 이미 처리한 것일 수 있습니다. 트러스트된 CA가 사용되지 않으면 무시합니다.

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

3. ONTAP 클러스터에 클라이언트 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
```

참고	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 인증서 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 테스트합니다. ONTAP 관리 LIF> 및 <SVM 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다.

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

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

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

```
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

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

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

인증 방법을 업데이트하거나 자격 증명을 회전합니다

다른 인증 방법을 사용하거나 자격 증명을 회전하도록 기존 백엔드를 업데이트할 수 있습니다. 이렇게 하면 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하도록 업데이트할 수 있고 인증서를 사용하는 백엔드는 사용자 이름/암호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 필요한 매개 변수가 포함된 업데이트된 backend.json 파일을 사용하여 'tridentctl backend update'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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

참고	암호를 회전할 때 스토리지 관리자는 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 다음에는 백엔드 업데이트가 있습니다. 인증서를 회전할 때 여러 인증서를 사용자에게 추가할 수 있습니다. 그런 다음 백엔드가 업데이트되어 새 인증서를 사용합니다. 그러면 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스가 중단되거나 이후에 생성된 볼륨 연결에 영향을 미치지 않습니다. 백엔드 업데이트에 성공하면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident에 대한 사용자 지정 ONTAP 역할을 생성합니다

Privileges에서 작업을 수행할 때 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 Trident로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident은 사용자가 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 을 "Trident 사용자 지정 역할 생성기"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 사용자에 대한 사용 이름 만들기:

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

3. 역할을 사용자에게 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 레벨에서 사용자 지정 역할을 생성하려면 * 클러스터 > 설정 * 을 선택합니다.

      SVM 레벨에서 사용자 지정 역할을 생성하려면 * 스토리지 > 스토리지 VM >> 설정 > 사용자 및 역할 * 을 선택합니다 `required SVM`.

   b. 사용자 및 역할 * 옆의 화살표 아이콘(*→ *)을 선택합니다.
   c. 역할 * 아래에서 * + 추가 * 를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 * 저장 * 을 클릭합니다.
2. * 역할을 Trident 사용자에게 매핑 *: + * 사용자 및 역할 * 페이지에서 다음 단계를 수행하십시오.
   a. 사용자 * 아래에서 추가 아이콘 * + * 를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 * Role * 에 대한 드롭다운 메뉴에서 역할을 선택합니다.
   c. 저장 * 을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- "ONTAP 관리를 위한 사용자 지정 역할" 또는 "사용자 지정 역할을 정의합니다"
- "역할 및 사용자 작업"

양방향 **CHAP**를 사용하여 연결 인증

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

경고 useCHAP는 한 번만 설정할 수 있는 Boolean 옵션이다. 기본적으로 false로 설정되어 있습니다. true 로 설정한 후에는 false 로 설정할 수 없습니다.

useCHAP=true 외에, chapInitatorSecret, chapTargetInitialatorSecret, chapchTargetUsername, chapUsername 필드가 백엔드 정의에 포함되어야 합니다. tridentctl update를 실행하여 백엔드를 생성한 후 비밀을 변경할 수 있다.

작동 원리

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

- SVM에서 CHAP 설정:
  - SVM의 기본 이니시에이터 보안 유형이 NONE(기본값 설정) * 이고 * 이미 존재하는 LUN이 볼륨에 없는 경우 Trident는 기본 보안 유형을 로 설정하고 CHAP 이니시에이터 및 타겟 사용자 이름과 암호를 구성합니다. CHAP
  - SVM에 LUN이 포함된 경우 Trident은 SVM에서 CHAP를 사용하도록 설정하지 않습니다. 따라서 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 생성된 후 Trident는 해당 tridentbackend CRD를 생성하고 CHAP 암호 및 사용자 이름을 Kubernetes 비밀로 저장합니다. 이 백엔드에서 Trident에 의해 생성된 모든 PVS가 CHAP를 통해 마운트되고 연결됩니다.

자격 증명을 순환하고 백엔드를 업데이트합니다.

backend.json 파일에서 CHAP 파라미터를 업데이트하여 CHAP 자격 증명을 업데이트할 수 있다. 이렇게 하려면 CHAP 암호를 업데이트하고 "tridentctl update" 명령을 사용하여 이러한 변경 사항을 반영해야 합니다.

경고 백엔드의 CHAP 암호를 업데이트할 때 을 사용하여 백엔드를 업데이트해야 tridentctl 합니다. Trident은 이러한 변경 사항을 파악할 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 않고 그대로 유지되며 SVM의 Trident가 자격 증명을 업데이트하는 경우 계속 활성 상태로 유지됩니다. 새 연결은 업데이트된 자격 증명을 사용하며 기존 연결은 계속 활성 상태를 유지합니다. 기존 PVS를 연결 해제하고 다시 연결하면 업데이트된 자격 증명을 사용하게 됩니다.

ONTAP SAN 구성 옵션 및 예

Trident 설치와 함께 ONTAP SAN 드라이버를 생성하고 사용합니다. 이 섹션에서는 백엔드를 StorageClasses에 매핑하기 위한 백엔드 구성 예제와 세부 정보를 제공합니다. "ASA r2 시스템"은(는) 스토리지 계층 구현 방식에서 다른 ONTAP 시스템(ASA, AFF, FAS)과 다릅니다. 이러한 차이점은 아래 표에 표시된 특정 매개변수 사용에 영향을 미칩니다. "ASA r2 시스템과 다른 ONTAP 시스템 간의 차이점에 대해 자세히 알아보세요.". Trident 백엔드 구성에서 시스템이 ASA r2임을 명시적으로 지정할 필요는 없습니다. `ontap-san`을(를) `storageDriverName`로 선택하면 Trident가 ASA r2 또는 다른 ONTAP 시스템을 자동으로 감지합니다. 아래 표에 표시된 것처럼 일부 백엔드 구성 매개변수는 ASA r2 시스템에는 적용되지 않습니다.

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

백엔드 구성 옵션

26.06 릴리스부터 ONTAP 균형 배치 기능을 활성화하여 ONTAP이 새 볼륨에 대한 애그리게이트를 선택하도록 할 수 있습니다. 요구 사항 및 예제는 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오.

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값
'내전'		항상 1

매개 변수	설명	기본값
'torageDriverName'입니다	스토리지 드라이버의 이름입니다. 생성 후에는 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`ontap-san` 또는 `ontap-san-economy`
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF
마나멘타LIF	클러스터 또는 SVM 관리 LIF의 IP 주소입니다. FQDN(정규화된 도메인 이름)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident가 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 과 같이 대괄호로 정의해야 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 합니다. 원활한 MetroCluster 전환은 를 MetroCluster 예참조하십시오. 참고 "vsadmin" 자격 증명을 사용하는 경우, 은 `managementLIF` SVM의 자격 증명이어야 합니다. "admin" 자격 증명을 사용하는 경우에는 이 클러스터의 자격 증명이어야 `managementLIF` 합니다.	"10.0.0.1","[2001:1234:ABCD::fefe]"
다타LIF	프로토콜 LIF의 IP 주소입니다. IPv6 플래그를 사용하여 Trident가 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 과 같이 대괄호로 정의해야 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 합니다. **iSCSI**에 대해 지정하지 마십시오**.** Trident는 을 사용하여 "ONTAP 선택적 LUN 맵"다중 경로 세션을 설정하는 데 필요한 iSCSI LIF를 검색합니다. 이 명시적으로 정의된 경우 경고가 `dataLIF` 생성됩니다. * MetroCluster의 경우 생략합니다. * 를 MetroCluster 예참조하십시오.	SVM에서 파생됩니다
'VM'입니다	사용할 스토리지 가상 머신입니다 *MetroCluster의 경우 생략합니다. * 를 참조하십시오 MetroCluster 예.	SVM 'managementLIF'가 지정된 경우에 파생됩니다
'useCHAP'입니다	CHAP를 사용하여 ONTAP SAN 드라이버에 대한 iSCSI 인증 [Boolean]. 백엔드에 제공된 SVM에 대한 기본 인증으로 양방향 CHAP를 구성하고 사용하려면 Trident에 대해 으로 `true` 설정합니다. 자세한 내용은 을 "ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 합니다" 참조하십시오. **FCP** 또는 **NVMe/TCP**에서는 지원되지 않습니다**.**	거짓입니다
챕터시크릿	CHAP 이니시에이터 암호입니다. useCHAP=true인 경우 필수입니다	""
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""

매개 변수	설명	기본값
챕터타겟이니터시크릿	CHAP 타겟 이니시에이터 암호입니다. useCHAP=true인 경우 필수입니다	""
'chapUsername' 입니다	인바운드 사용자 이름입니다. useCHAP=true인 경우 필수입니다	""
'chapTargetUsername'입니다	대상 사용자 이름입니다. useCHAP=true인 경우 필수입니다	""
'고객증명서'	Base64 - 클라이언트 인증서의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
'clientPrivateKey' 입니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
신탁인증서다	Base64 - 신뢰할 수 있는 CA 인증서의 인코딩된 값입니다. 선택 사항. 인증서 기반 인증에 사용됩니다.	""
'사용자 이름'	ONTAP 클러스터와 통신하려면 사용자 이름이 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
"암호"	ONTAP 클러스터와 통신하려면 비밀번호가 필요합니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	""
'VM'입니다	사용할 스토리지 가상 머신입니다	SVM 'managementLIF'가 지정된 경우에 파생됩니다
'토르agePrefix'	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사입니다. 나중에 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`trident`

매개 변수	설명	기본값
골재	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. 참고: SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. 볼륨을 프로비저닝하기 위해 Trident에서 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 이동되면 SVM 애그리게이트를 폴링하는 동안 백엔드가 Trident에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **ASA r2** 시스템에 대해서는 지정하지 마세요. • `useBalancedPlacement`이(가) `true`(으)로 설정된 경우 지정하지 마십시오.*	""
`useBalancedPlacement`	새 볼륨에 대한 애그리게이트 선택을 ONTAP에 위임합니다[Boolean]. ONTAP 9.19.1 이상이 필요하며 `useREST`을(를) `true`로 설정해야 합니다. `aggregate`와 함께 사용할 수 없습니다. 요구 사항 및 예시는 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오.	거짓입니다
제한선태사용법	사용량이 이 비율을 초과하면 프로비저닝이 실패합니다. Amazon FSx for NetApp ONTAP 백엔드를 사용하는 경우 을 지정하지 `limitAggregateUsage`마십시오. 제공된 및 `vsadmin` 에는 `fsxadmin` 애그리게이트 사용량을 검색하고 Trident를 사용하여 제한하는 데 필요한 권한이 포함되어 있지 않습니다. **ASA r2** 시스템에 대해서는 지정하지 마세요.	""(기본적으로 적용되지 않음)
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다. 또한 LUN에 대해 관리하는 볼륨의 최대 크기를 제한합니다.	""(기본적으로 적용되지 않음)
'오만유연한'	FlexVol당 최대 LUN 수는 범위[50, 200]에 있어야 합니다.	`100`
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true} 문제 해결 중이지 않고 자세한 로그 덤프가 필요한 경우가 아니면 사용하지 마십시오.	`null`

매개 변수	설명	기본값
'useREST'	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. `useREST`설정 시 `true` , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. `false` Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. `ontapi` 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. `vsadmin` 그리고 `cluster-admin` 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST` 로 설정됩니다 `true` 기본적으로; 변경 `useREST` 에게 `false` ONTAPI(ZAPI) 호출을 사용합니다. `useREST`NVMe/TCP에 완벽하게 적합합니다. 참고 NVMe는 ONTAP REST API에서만 지원되며 ONTAPI(ZAPI)에서는 지원되지 않습니다. *지정된 경우 항상 다음으로 설정됩니다. true ASA r2 시스템*용.	true ONTAP 9.15.1 이상, 그렇지 않은 경우 false.
sanType	iSCSI, nvme NVMe/TCP 또는 fcp FC(SCSI over Fibre Channel)를 선택할 때 iscsi 사용합니다.	iscsi 비어 있는 경우
formatOptions	`formatOptions`을 사용하여 `mkfs` 명령에 대한 명령줄 인수를 지정합니다. 이 인수는 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" 지원됨 **ontap-san** 그리고 **ontap-san-economy** **iSCSI** 프로토콜을 사용하는 드라이버. 또한 **iSCSI** 및 **NVMe/TCP** 프로토콜을 사용하는 **ASA r2** 시스템에서도 지원됩니다.	

매개 변수	설명	기본값
`limitVolumePoolSize`	ONTAP-SAN-Economy 백엔드에서 LUN을 사용할 때 요청될 수 있는 최대 FlexVol 크기입니다.	""(기본적으로 적용되지 않음)
`denyNewVolumePools`	백엔드가 LUN을 포함하도록 새 FlexVol 볼륨을 생성하지 못하도록 `ontap-san-economy` 제한합니다. 기존 FlexVol만 새 PVS 프로비저닝에 사용됩니다.	

포맷옵션 사용에 대한 권장 사항

Trident 포맷 과정을 신속하게 진행하기 위해 다음 옵션을 권장합니다.

- **-E nodiscard(ext3, ext4):** mkfs 시점에 블록을 삭제하려고 시도하지 마세요(처음에 블록을 삭제하는 것은 솔리드 스테이트 장치와 스파스/씬 프로비저닝 스토리지에서 유용합니다). 이는 더 이상 사용되지 않는 옵션인 "-K"를 대체하며 ext3, ext4 파일 시스템에 적용할 수 있습니다.
- **-K (xfs):** mkfs 시간에 블록을 버리려고 시도하지 마십시오. 이 옵션은 xfs 파일 시스템에 적용할 수 있습니다.

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

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

단계

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

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

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

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

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

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

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

볼륨 프로비저닝을 위한 백엔드 구성 옵션

에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다 `defaults` 섹션을 참조하십시오. 예를 들어, 아래 구성 예제를 참조하십시오.

매개 변수	설명	기본값
'팩시배부	LUN에 대한 공간 할당	"true" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
'예비공간'	공간 예약 모드, "없음"(씬) 또는 "볼륨"(일반). 설정 **none** **ASA r2** 시스템용.	"없음"
냅샷정책	사용할 스냅샷 정책입니다. *설정 `none` ASA r2 시스템*용.	"없음"
"qosPolicy"	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9 .8 이상이 필요합니다. 비공유 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 제한을 적용합니다.	""
적응성 QosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다	""
안산예비역	스냅숏용으로 예약된 볼륨의 비율입니다. **ASA r2** 시스템에 대해서는 지정하지 마세요.	"0"인 경우 `snapshotPolicy` "없음"이고, 그렇지 않으면""입니다.
'plitOnClone'을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	"거짓"
암호화	NetApp 볼륨 암호화(NVE)를 새 볼륨에서 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이선스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방법".	"false" *지정된 경우 설정됨 `true` ASA r2 시스템*용.
`luksEncryption`	LUKS 암호화를 사용합니다. 을 "LUKS(Linux Unified Key Setup) 사용"참조하십시오.	"" *로 설정 `false` ASA r2 시스템*용.
'계층화 정책'	"없음"을 사용하는 계층화 정책 **ASA r2** 시스템에는 지정하지 마세요.	
`nameTemplate`	사용자 지정 볼륨 이름을 생성하는 템플릿입니다.	""

볼륨 프로비저닝의 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'
```

참고

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.

을 정의하는 백엔드의 경우 snapshotReserve Trident는 다음과 같이 볼륨 크기를 계산합니다.

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

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



현재 기존 볼륨에 대해 새 계산을 사용하는 유일한 방법은 크기 조정입니다.

최소 구성의 예

다음 예에서는 대부분의 매개 변수를 기본값으로 두는 기본 구성을 보여 줍니다. 이는 백엔드를 정의하는 가장 쉬운 방법입니다.

참고 | NetApp ONTAP on Trident와 함께 Amazon FSx를 사용하는 경우, NetApp은 IP 주소 대신 LIF에 대한 DNS 이름을 지정할 것을 권장합니다.

▼ **ONTAP SAN**의 예

이것은 를 사용하는 기본 구성입니다 `ontap-san` 드라이버.

```
---
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** 예

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트할 필요가 없도록 백엔드를 구성할 수 있습니다 "SVM 복제 및 복구".

원활한 스위치오버 및 스위치백의 경우 를 사용하여 SVM을 지정하고 `managementLIF` 매개 변수를 생략합니다. `svm` 예를 들면 다음과 같습니다.

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

▼ **ONTAP SAN** 경제 예

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

▼ 인증서 기반 인증의 예

이 기본 구성 예에서 `clientCertificate`, `clientPrivateKey`, 및 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 에 채워집니다 `backend.json` 그리고 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 예에서는 를 사용하여 백엔드를 생성합니다 `useCHAP` 를 로 설정합니다 `true`.

ONTAP SAN CHAP의 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 이코노미 **CHAP**의 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>
```

▼ 균형 잡힌 배치 예시

이 예제는 ONTAP 균형 배치 기능을 활성화하여 ONTAP이 새 볼륨에 대한 애그리게이트를 선택하도록 합니다. 균형 배치 기능을 사용하려면 ONTAP 9.19.1 이상 버전이 필요합니다. 자세한 내용은 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오.

```
---
version: 1
storageDriverName: ontap-san
backendName: san-balanced-placement
managementLIF: 10.0.0.1
svm: svm_iscsi
username: vsadmin
password: <password>
useBalancedPlacement: true
```

▼ **NVMe/TCP** 예

ONTAP 백엔드에서 NVMe로 구성된 SVM이 있어야 합니다. NVMe/TCP에 대한 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
```

▼ **FCP(SCSI over FC)** 예

ONTAP 백엔드에서 FC로 SVM을 구성해야 합니다. FC에 대한 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
```

▼ **nameTemplate**이 포함된 백엔드 구성 예

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

▼ **ONTAP-SAN**-이코노미 드라이버에 대한 옵션 예

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

가상 풀의 백엔드 예

이러한 백엔드 정의 파일 샘플에서는 와 같은 모든 스토리지 풀에 대해 특정 기본값이 설정됩니다 `spaceReserve` 없음, `spaceAllocation` 거짓일 경우, 및 `encryption` 거짓일 때. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident는 "Comments" 필드에 프로비저닝 레이블을 설정합니다. FlexVol volume Trident에 주석이 설정된 용량 할당 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨으로 복제합니다. 편의를 위해 스토리지 관리자는 가상 풀 및 그룹 볼륨별로 레이블을 레이블별로 정의할 수 있습니다.

이 예에서는 일부 스토리지 풀이 자체적으로 설정됩니다 `spaceReserve`, `spaceAllocation`, 및 `encryption` 일부 풀은 기본값을 재정의합니다.

▼ **ONTAP SAN**의 예

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

```
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

▼ **ONTAP SAN** 경제 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 풀의 백엔드 예을 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 이제 볼륨은 선택된 가상 풀에 정의된 속성을 갖게 됩니다.

- The `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"
```

- The `protection-not-gold` StorageClass는 `ontap-san` 백엔드의 두 번째 및 세 번째 가상 풀에 매핑됩니다. 이 풀들만이 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"
```

- The `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"
```

- The `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"
```

- The `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"
```

- The `my-test-app-sc` StorageClass는 `testAPP` `ontap-san`드라이버의 가상 풀에 `sanType: nvme`로 매핑됩니다. 이것이 유일하게 `testApp`를 제공하는 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 충족되는지 확인합니다.

NVMe 데이터 **LIF** 새로 고침

백엔드를 다시 생성하거나 Trident를 다시 시작할 필요 없이 게시된 NVMe 볼륨이 추가, 제거 또는 새 주소가 할당된 LIF를 사용할 수 있도록 NVMe 데이터 LIF 새로 고침을 활성화합니다.

개요

ONTAP SAN 백엔드를 추가하면 Trident는 SVM에 대한 NVMe 데이터 LIF를 검색합니다. 기본적으로 Trident는 백엔드를 다시 추가하거나 Trident를 다시 시작할 때까지 해당 LIF 세트를 유지합니다.

Trident가 실행 중일 때 NVMe 데이터 LIF를 새로 고치려면 `enableDataLIFRefresh`을 (를) `true`(으)로 설정합니다. 그러면 게시된 NVMe 볼륨은 추가, 제거 또는 새 주소가 할당된 LIF를 사용합니다.

이 옵션은 설치 매개변수이며 백엔드 구성 매개변수가 아닙니다. 기본적으로 비활성화되어 있습니다.

NVMe 데이터 LIF 새로 고침 기능은 Trident 26.10 버전부터 사용할 수 있습니다.

이 기능은 다음 항목에 적용됩니다:

- `ontap-san`(통합 ONTAP에서 `sanType: nvme`와(과) 함께).
- `ontap-san` ASA r2에서 NVMe/TCP를 사용합니다.

주의	모든 데이터 LIF를 동시에 제거하거나 교체하면 시스템에 장애가 발생할 수 있습니다. 모든 데이터 LIF를 제거하거나 교체할 계획이라면 데이터 액세스가 중단되지 않도록 먼저 대체 LIF를 설치해야 합니다.

NVMe 데이터 **LIF** 새로 고침 활성화

Trident 운영자

새로 설치하는 경우 `TridentOrchestrator` spec에서 `enableDataLIFRefresh`을 설정하십시오.

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

매니페스트를 적용합니다:

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

기존 설치의 경우 `TridentOrchestrator`을(를) 패치하십시오:

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

적용된 값을 확인하십시오:

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

헬름

새 설치의 경우 Helm 값을 설정하십시오:

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

기존 Helm 설치의 경우, 값을 활성화하여 릴리스를 업그레이드하십시오:

```
helm upgrade trident netapp-trident/trident-operator \
  --namespace trident \
  --reuse-values \
  --set enableDataLIFRefresh=true
```

Helm 값 파일에서 `enableDataLIFRefresh: true`을 설정하고 `-f <values-file>`을 사용하여 `helm upgrade`에 전달할 수도 있습니다.

tridentctl 을 선택합니다

새 설치의 경우 `--enable-datalif-refresh` 플래그를 사용하십시오:

```
tridentctl install --enable-datalif-refresh -n trident
```

기존 `tridentctl` 설치에서 옵션을 활성화하려면 Trident를 제거하고 플래그를 사용하여 다시 설치하십시오. 제거하기 전에 "Trident를 제거합니다"의 주의 사항을 따르십시오.

```
tridentctl uninstall -n trident
tridentctl install --enable-datalif-refresh -n trident
```

참고	`--enable-datalif-refresh` 플래그는 설치 옵션이며 `tridentctl install --help`에 표시되지 않습니다.

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 정보입니다

Trident는 ONTAP 클러스터와 통신할 수 있도록 다음과 같은 NAS 스토리지 드라이버를 제공합니다. 지원되는 액세스 모드는 *ReadWriteOnce*(RWO), *ReadOnlyMany*(ROX), *ReadWriteMany*(rwx), *ReadWriteOncePod*(RWOP)입니다.

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'ONTAP-NAS'	NFS 를 참조하십시오 중소기업	파일 시스템	RWO, ROX, rwx, RWOP	"", `nfs`, `smb`
ONTAP-NAS-이코노미	NFS 를 참조하십시오 중소기업	파일 시스템	RWO, ROX, rwx, RWOP	"", `nfs`, `smb`

드라이버	프로토콜	볼륨 모드	액세스 모드가 지원됩니다	지원되는 파일 시스템
'ONTAP-NAS-Flexgroup'	NFS 를 참조하십시오 중소기업	파일 시스템	RWO, ROX, rwx, RWOP	"", `nfs`, `smb`

경고

- 사용 `ontap-san-economy` 영구 볼륨 사용 수가 보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한".
- 사용 `ontap-nas-economy` 영구 볼륨 사용 수가 보다 높을 것으로 예상되는 경우에만 "지원되는 ONTAP 볼륨 제한" 및 `ontap-san-economy` 드라이버를 사용할 수 없습니다.
- 사용하지 마십시오 `ontap-nas-economy` 데이터 보호, 재해 복구 또는 이동성이 필요할 것으로 예상되는 경우
- ONTAP 볼륨 자동 크기 조정은 ONTAP NAS 드라이버에서 지원되지 않습니다. 지원되는 SAN 드라이버 및 구성 세부 정보는 "ONTAP 볼륨 자동 크기 조정을 구성합니다"를 참조하십시오.

사용자 권한

Trident는 ONTAP 또는 SVM 관리자로 실행해야 하며, 일반적으로 클러스터 사용자 `vsadmin` 또는 SVM 사용자 또는 같은 역할을 가진 다른 이름의 사용자를 사용할 `admin` 것입니다.

Amazon FSx for NetApp ONTAP 배포의 경우 Trident은 클러스터 사용자 또는 `vsadmin` SVM 사용자를 사용하여 ONTAP 또는 SVM 관리자로 실행하거나 `fsxadmin` 동일한 역할을 가진 다른 이름의 사용자를 실행해야 합니다. `fsxadmin`사용자는 클러스터 관리자를 제한적으로 대체합니다.

참고

`limitAggregateUsage`매개 변수를 사용하려면 클러스터 관리 권한이 필요합니다. `Trident`와 함께 `Amazon FSx for NetApp ONTAP`를 사용할 때 `limitAggregateUsage` 매개 변수는 및 `fsxadmin` 사용자 계정에서 작동하지 `vsadmin` 않습니다. 이 매개 변수를 지정하면 구성 작업이 실패합니다.

Trident 드라이버가 사용할 수 있는 더 제한적인 역할을 ONTAP 내에 만들 수 있지만 권장하지 않습니다. Trident의 대부분의 새로운 릴리즈에서는 추가 API를 호출하므로 업그레이드가 어렵고 오류가 발생하기 쉽습니다.

ONTAP NAS 드라이버를 사용하여 백엔드를 구성할 준비를 합니다

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

참고

오직 `ontap-nas` 드라이버(NFS 프로토콜 포함)는 AFX 시스템에서 지원됩니다. SMB 프로토콜은 지원되지 않습니다.

요구 사항

- 모든 ONTAP 백엔드의 경우 Trident에서는 최소한 하나의 집계가 SVM에 할당되어야 합니다.

- 둘 이상의 드라이버를 실행하고 둘 중 하나를 가리키는 스토리지 클래스를 생성할 수 있습니다. 예를 들어, 을 사용하는 Gold 클래스를 구성할 수 있습니다 `ontap-nas` 드라이버 및 를 사용하는 Bronze 클래스 `ontap-nas-economy` 1개.
- 모든 Kubernetes 작업자 노드에 적절한 NFS 툴이 설치되어 있어야 합니다. 을 참조하십시오 "여기" 를 참조하십시오.
- Trident는 Windows 노드에서만 실행되는 Pod에 마운트된 SMB 볼륨을 지원합니다. 자세한 내용은 을 SMB 볼륨 프로비저닝을 위한 준비 참조하십시오.

ONTAP 백엔드를 인증합니다

Trident는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- 자격 증명 기반: 이 모드에서는 ONTAP 백엔드에 대한 충분한 권한이 필요합니다. 과 같이 미리 정의된 보안 로그인 역할과 연결된 계정을 사용하는 것이 좋습니다 `admin` 또는 `vsadmin` ONTAP 버전과의 호환성을 최대한 보장하기 위해
- 인증서 기반: 이 모드에서는 Trident가 ONTAP 클러스터와 통신하기 위해 백엔드에 인증서가 설치되어 있어야 합니다. 이 경우 백엔드 정의에는 클라이언트 인증서, 키 및 사용할 경우 신뢰할 수 있는 CA 인증서의 Base64로 인코딩된 값이 있어야 합니다(권장).

자격 증명 기반 방법과 인증서 기반 방법 간에 이동하기 위해 기존 백엔드를 업데이트할 수 있습니다. 그러나 한 번에 하나의 인증 방법만 지원됩니다. 다른 인증 방법으로 전환하려면 백엔드 구성에서 기존 방법을 제거해야 합니다.

경고	자격 증명과 인증서 * 를 모두 제공하려고 하면 구성 파일에 둘 이상의 인증 방법이 제공된다는 오류가 발생하여 백엔드 생성이 실패합니다.

자격 증명 기반 인증을 사용합니다

Trident은 ONTAP 백엔드와 통신하기 위해 SVM 범위/클러스터 범위 관리자에 대한 자격 증명이 필요합니다. 또는 `vsadmin` 과 같은 미리 정의된 표준 역할을 사용하는 것이 좋습니다 `admin`. 따라서 향후 Trident 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리즈와의 호환성이 보장됩니다. 사용자 지정 보안 로그인 역할을 만들어 Trident와 함께 사용할 수 있지만 권장하지는 않습니다.

백엔드 정의의 예는 다음과 같습니다.

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
```

JSON을 참조하십시오

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

백엔드 정의는 자격 증명이 일반 텍스트로 저장되는 유일한 위치라는 점에 유의하십시오. 백엔드가 생성된 후 사용자 이름/암호는 Base64로 인코딩되어 Kubernetes 암호로 저장됩니다. 백엔드의 생성/업딩은 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes/스토리지 관리자가 수행할 수 있는 관리 전용 작업입니다.

인증서 기반 인증을 사용합니다

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

- clientCertificate: Base64로 인코딩된 클라이언트 인증서 값입니다.
- clientPrivateKey: Base64 - 연결된 개인 키의 인코딩된 값입니다.
- TrustedCACertificate: 신뢰할 수 있는 CA 인증서의 Base64 인코딩 값입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

```
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. 신뢰할 수 있는 CA 인증서를 ONTAP 클러스터에 추가합니다. 이는 스토리지 관리자가 이미 처리한 것일 수 있습니다. 트러스트된 CA가 사용되지 않으면 무시합니다.

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

3. ONTAP 클러스터에 클라이언트 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 인증서 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 테스트합니다. ONTAP 관리 LIF> 및 <SVM 이름>을 관리 LIF IP 및 SVM 이름으로 바꿉니다. 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. Base64로 인증서, 키 및 신뢰할 수 있는 CA 인증서를 인코딩합니다.

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

7. 이전 단계에서 얻은 값을 사용하여 백엔드를 생성합니다.

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

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

인증 방법을 업데이트하거나 자격 증명을 회전합니다

다른 인증 방법을 사용하거나 자격 증명을 회전하도록 기존 백엔드를 업데이트할 수 있습니다. 이렇게 하면 사용자 이름/암호를 사용하는 백엔드를 인증서를 사용하도록 업데이트할 수 있고 인증서를 사용하는 백엔드는 사용자 이름/암호 기반으로 업데이트할 수 있습니다. 이렇게 하려면 기존 인증 방법을 제거하고 새 인증 방법을 추가해야 합니다. 그런 다음 실행할 필수 매개 변수가 포함된 업데이트된 backend.json 파일을 사용합니다 `tridentctl update backend`.

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

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

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
```

```
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

참고	암호를 회전할 때 스토리지 관리자는 먼저 ONTAP에서 사용자의 암호를 업데이트해야 합니다. 그 다음에는 백엔드 업데이트가 있습니다. 인증서를 회전할 때 여러 인증서를 사용자에게 추가할 수 있습니다. 그런 다음 백엔드가 업데이트되어 새 인증서를 사용합니다. 그러면 ONTAP 클러스터에서 이전 인증서를 삭제할 수 있습니다.

백엔드를 업데이트해도 이미 생성된 볼륨에 대한 액세스가 중단되거나 이후에 생성된 볼륨 연결에 영향을 미치지 않습니다. 백엔드 업데이트에 성공하면 Trident가 ONTAP 백엔드와 통신하여 향후 볼륨 작업을 처리할 수 있음을 나타냅니다.

Trident에 대한 사용자 지정 ONTAP 역할을 생성합니다

Privileges에서 작업을 수행할 때 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 Trident로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident은 사용자가 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 을 "Trident 사용자 지정 역할 생성기"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 사용자에 대한 사용 이름 만들기:

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

3. 역할을 사용자에게 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 레벨에서 사용자 지정 역할을 생성하려면 * 클러스터 > 설정 * 을 선택합니다.

SVM 레벨에서 사용자 지정 역할을 생성하려면 * 스토리지 > 스토리지 VM >> 설정 > 사용자 및 역할 * 을 선택합니다 `required SVM`.

b. 사용자 및 역할 * 옆의 화살표 아이콘(*→ *)을 선택합니다.

c. 역할 * 아래에서 * + 추가 * 를 선택합니다.

d. 역할에 대한 규칙을 정의하고 * 저장 * 을 클릭합니다.

2. * 역할을 Trident 사용자에게 매핑 *: + * 사용자 및 역할 * 페이지에서 다음 단계를 수행하십시오.

a. 사용자 * 아래에서 추가 아이콘 * + * 를 선택합니다.

b. 필요한 사용자 이름을 선택하고 * Role * 에 대한 드롭다운 메뉴에서 역할을 선택합니다.

c. 저장 * 을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- "ONTAP 관리를 위한 사용자 지정 역할" 또는 "사용자 지정 역할을 정의합니다"
- "역할 및 사용자 작업"

NFS 엑스포트 정책을 관리합니다

Trident는 NFS 엑스포트 정책을 사용하여 프로비저닝한 볼륨에 대한 액세스를 제어합니다.

Trident는 내보내기 정책을 사용할 때 두 가지 옵션을 제공합니다.

- Trident는 엑스포트 정책 자체를 동적으로 관리할 수 있습니다. 이 운영 모드에서 스토리지 관리자는 허용되는 IP 주소를 나타내는 CIDR 블록의 목록을 지정합니다. Trident는 이러한 범위에 속하는 적용 가능한 노드 IP를 게시 시 자동으로 내보내기 정책에 추가합니다. 또는 CIDR을 지정하지 않으면 게시되는 볼륨이 있는 노드에서 찾은 모든 글로벌 범위 유니캐스트 IP가 엑스포트 정책에 추가됩니다.
- 스토리지 관리자는 엑스포트 정책을 생성하고 규칙을 수동으로 추가할 수 있습니다. 구성에 다른 엑스포트 정책 이름을 지정하지 않는 한 Trident는 기본 엑스포트 정책을 사용합니다.

엑스포트 정책을 동적으로 관리

Trident는 ONTAP 백엔드에 대한 엑스포트 정책을 동적으로 관리하는 기능을 제공합니다. 따라서 스토리지 관리자는 명시적 규칙을 수동으로 정의하는 대신 작업자 노드 IP에 허용되는 주소 공간을 지정할 수 있습니다. 엑스포트 정책 관리를 크게 간소화하므로, 엑스포트 정책을 수정하면 더 이상 스토리지 클러스터에 대한 수동 작업이 필요하지 않습니다. 또한 이렇게 하면 볼륨을 마운트하고 지정된 범위 내에서 IP를 갖는 작업자 노드만 스토리지 클러스터에 대한 액세스를 제한하여 세분화된 자동 관리를 지원합니다.

참고	동적 내보내기 정책을 사용할 때는 NAT(Network Address Translation)를 사용하지 마십시오. NAT를 사용하면 스토리지 컨트롤러는 실제 IP 호스트 주소가 아니라 프런트엔드 NAT 주소를 인식하므로 내보내기 규칙에 일치하는 항목이 없으면 액세스가 거부됩니다.

예

두 가지 구성 옵션을 사용해야 합니다. 다음은 백엔드 정의의 예입니다.

```
---
```

```
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
```

참고	이 기능을 사용할 때는 SVM의 루트 교차점에 노드 CIDR 블록(예: 기본 엑스포트 정책)을 허용하는 엑스포트 규칙과 함께 이전에 생성된 엑스포트 정책이 있는지 확인해야 합니다. Trident 전용 SVM을 사용하려면 항상 NetApp 권장 모범 사례를 따르십시오.

다음은 위의 예를 사용하여 이 기능이 작동하는 방식에 대한 설명입니다.

- `autoExportPolicy` 가 로 설정되어 `true` 있습니다. 이는 Trident이 SVM에 대해 이 백엔드로 프로비저닝된 각 볼륨에 대한 엑스포트 정책을 `svm1` 생성하고 주소 블록을 사용하여 규칙 추가 및 삭제를 `autoexportCIDRs` 처리합니다. 볼륨이 노드에 연결될 때까지 볼륨은 규칙 없이 빈 엑스포트 정책을 사용하여 볼륨에 대한 원치 않는 액세스를 차단합니다. 볼륨이 노드에 게시되면 Trident에서 지정된 CIDR 블록 내에 노드 IP를 포함하는 기본 qtree와 같은 이름의 엑스포트 정책을 생성합니다. 이러한 IP는 상위 FlexVol volume에서 사용하는 내보내기 정책에도 추가됩니다
  - 예를 들면 다음과 같습니다.
    - 백엔드 UUID 403b5326-8482-40dB-96d0-d83fb3f4daec
    - `autoExportPolicy` 로 설정합니다 `true`
    - 스토리지 접두사입니다 `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - 이름이 Trident_PVC_a79bcf5f_7b6d_4a40_9876_e2551f159c1c인 FlexVol qtree에 대한 엑스포트 정책, 이름이 인 qtree에 대한 `trident-403b5326-8482-40db96d0-d83fb3f4daec` 엑스포트 정책, `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` SVM에 명명된 빈 엑스포트 정책을 `trident_empty` 생성합니다. FlexVol 엑스포트 정책 규칙은 qtree 엑스포트 정책에 포함된 모든 규칙의 상위 집합이 됩니다. 빈 내보내기 정책은 연결되지 않은 모든 볼륨에서 다시 사용됩니다.
- `autoExportCIDRs` 주소 블록 목록을 포함합니다. 이 필드는 선택 사항이며 기본적으로 ["0.0.0.0/0", ":/0"] 입니다. 정의되지 않은 경우 Trident는 작업자 노드에 있는 모든 전역 범위의 유니캐스트 주소를 게시물과 함께 추가합니다.

이 예에서는 `192.168.0.0/24` 주소 공간이 제공됩니다. 이는 발행물이 있는 이 주소 범위에 속하는 Kubernetes 노드 IP가 Trident에서 생성하는 엑스포트 정책에 추가된다는 것을 나타냅니다. Trident는 실행되는 노드를 등록할 때 노드의 IP 주소를 검색하여 에서 제공하는 주소 블록과 대조하여 확인합니다 `autoExportCIDRs`. 게시 시 IP를 필터링한 후 Trident는 게시 대상 노드의 클라이언트 IP에 대한 내보내기 정책 규칙을 만듭니다.

백엔드를 생성한 후 백엔드에 대한 자동 내보내기 정책 및 자동 내보내기 CIDR을 업데이트할 수 있습니다. 기존 CIDR을 자동으로 관리하거나 삭제하는 백엔드에 새 CIDR을 추가할 수 있습니다. CIDR을 삭제할 때는 기존 연결이 끊어지지 않도록 주의해야 합니다. 백엔드에 대해 'autoExportPolicy'를 사용하지 않도록 설정하고 수동으로 생성된 내보내기 정책으로 돌아갈 수도 있습니다. 이렇게 하려면 백엔드 구성에서 'exportPolicy' 매개 변수를 설정해야 합니다.

Trident에서 백엔드를 생성하거나 업데이트한 후 또는 해당 `tridentbackend` CRD를 사용하여 백엔드를 확인할 수 `tridentctl` 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 노드에 해당하는 액세스 규칙을 제거합니다. Trident는 관리되는 백엔드의 내보내기 정책에서 이 노드 IP를 제거하여 클러스터의 새 노드에서 이 IP를 재사용하지 않는 한 불량 마운트를 방지합니다.

기존 백엔드의 경우 백엔드를 로 업데이트하면 `tridentctl update backend` Trident에서 엑스포트 정책을 자동으로 관리할 수 있습니다. 이렇게 하면 필요한 경우 백엔드의 UUID 및 qtree 이름을 따서 명명된 두 개의 새 엑스포트 정책이 생성됩니다. 백엔드에 있는 볼륨은 마운트 해제했다가 다시 마운트하면 새로 생성된 엑스포트 정책을 사용합니다.

참고	자동 관리되는 내보내기 정책이 있는 백엔드를 삭제하면 동적으로 생성된 내보내기 정책이 삭제됩니다. 백엔드가 다시 생성되면 백엔드가 새 백엔드로 처리되어 새 엑스포트 정책이 생성됩니다.

라이브 노드의 IP 주소가 업데이트되면 노드에서 Trident Pod를 다시 시작해야 합니다. 그런 다음 Trident는 이 IP 변경 사항을 반영하도록 관리하는 백엔드에 대한 내보내기 정책을 업데이트합니다.

SMB 볼륨 프로비저닝을 위한 준비

준비를 조금만 더 하면 를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다 `ontap-nas` 드라이버.

경고	ONTAP 온프레미스 클러스터를 위한 SMB 볼륨을 생성하려면 SVM에서 NFS 및 SMB/CIFS 프로토콜을 모두 구성해야 `ontap-nas-economy` 합니다. 이 두 프로토콜 중 하나를 구성하지 않으면 SMB 볼륨 생성에 실패합니다.

참고 `autoExportPolicy` SMB 볼륨에는 가 지원되지 않습니다.

시작하기 전에

SMB 볼륨을 프로비저닝하려면 먼저 다음 항목이 있어야 합니다.

- Linux 컨트롤러 노드 및 Windows Server 2022를 실행하는 Windows 작업자 노드가 있는 Kubernetes 클러스터 Trident는 Windows 노드에서만 실행되는 Pod에 마운트된 SMB 볼륨을 지원합니다.
- Active Directory 자격 증명이 포함된 Trident 암호가 하나 이상 있습니다. 비밀 생성하기 `smbcreds`:

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

- Windows 서비스로 구성된 CSI 프록시. 를 구성합니다 `csi-proxy`를 참조하십시오 "GitHub:CSI 프록시" 또는 "GitHub: Windows용 CSI 프록시" Windows에서 실행되는 Kubernetes 노드의 경우:

단계

1. 온프레미스 ONTAP의 경우 선택적으로 SMB 공유를 생성하거나 Trident에서 공유를 생성할 수 있습니다.

   참고 ONTAP용 Amazon FSx에는 SMB 공유가 필요합니다.

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

   a. 필요한 경우 공유에 대한 디렉토리 경로 구조를 생성합니다.

      를 클릭합니다 `vserver cifs share create` 명령은 공유를 생성하는 동안 -path 옵션에 지정된 경로를 확인합니다. 지정한 경로가 없으면 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
      ```

      참고 을 참조하십시오 "SMB 공유를 생성합니다" 를 참조하십시오.

2. 백엔드를 생성할 때 SMB 볼륨을 지정하려면 다음을 구성해야 합니다. 모든 ONTAP 백엔드 구성 옵션에 대한 자세한 내용은 을 참조하십시오 "ONTAP 구성 옵션 및 예제용 FSX".

매개 변수	설명	예
smbShare	Microsoft 관리 콘솔 또는 ONTAP CLI를 사용하여 생성된 SMB 공유의 이름, Trident에서 SMB 공유를 생성할 수 있는 이름, 볼륨에 대한 일반적인 공유 액세스를 방지하기 위해 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 사내 ONTAP의 경우 선택 사항입니다. 이 매개변수는 ONTAP 백엔드에 대한 아마존 FSx에 필요하며 비워둘 수 없습니다.	smb-share
nasType	* 를 로 설정해야 합니다 smb. * null인 경우 기본값은 로 설정됩니다 nfs.	smb
'생태성 스타일'을 참조하십시오	새로운 볼륨에 대한 보안 스타일 * 를 로 설정해야 합니다 ntfs 또는 mixed SMB 볼륨용. *	ntfs 또는 mixed SMB 볼륨용
유니크권한	모드를 선택합니다. SMB 볼륨에 대해서는 * 를 비워 두어야 합니다. *	""

보안 **SMB** 활성화

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

기억해야 할 사항

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

단계

1. 다음 예와 같이 TridentBackendConfig에 adAdminUser를 지정합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
```

```
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 저장 클래스에 주석을 추가합니다.

   추가하다 `trident.netapp.io/smbShareAdUser` 보안 SMB를 오류 없이 사용할 수 있도록 스토리지 클래스에 주석을 추가합니다. 주석에 지정된 사용자 값 `trident.netapp.io/smbShareAdUser` 사용자 이름에 지정된 것과 동일해야 합니다. `smbcreds` 비밀입니다. 다음 중 하나를 선택할 수 있습니다. `smbShareAdUserPermission` : `full_control` , `change` , 또는 `read` . `full_control` .

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

1. PVC을 생성합니다.

   다음 예제에서는 PVC를 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
```

```
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS 구성 옵션 및 예

Trident 설치에서 ONTAP NAS 드라이버를 생성하고 사용하십시오. 이 섹션에서는 백엔드 구성 예제와 백엔드를 StorageClasses에 매핑하는 방법에 대한 세부 정보를 제공합니다. 25.10 릴리스부터 NetApp Trident는 "NetApp AFX 스토리지 시스템"을 지원합니다. NetApp AFX 스토리지 시스템은 스토리지 계층 구현 방식에서 다른 ONTAP 기반 시스템(ASA, AFF, FAS)과 다릅니다.

참고	오직 `ontap-nas` 드라이버(NFS 프로토콜 포함)는 NetApp AFX 시스템에서 지원됩니다. SMB 프로토콜은 지원되지 않습니다.

백엔드 구성 옵션

Trident 백엔드 구성에서 시스템이 NetApp AFX 스토리지 시스템임을 지정할 필요가 없습니다. `ontap-nas`을 (를) `storageDriverName`(으)로 선택하면 Trident가 AFX 스토리지 시스템을 자동으로 감지합니다. 일부 백엔드 구성 매개변수는 AFX 스토리지 시스템에는 적용되지 않습니다.

26.06 릴리스부터 ONTAP 균형 배치 기능을 활성화하여 ONTAP이 새 볼륨에 대한 애그리게이트를 선택하도록 할 수 있습니다. 균형 배치 기능은 ONTAP 9.19.1 이상 버전의 AFX 스토리지 시스템에 필요합니다. 요구 사항 및 예제는 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오.

다음 표는 백엔드 구성 옵션을 보여줍니다.

매개 변수	설명	기본값
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다. 생성 후에는 수정할 수 없습니다. 이 매개변수를 업데이트하려면 새 백엔드를 생성해야 합니다. 참고: NetApp AFX 시스템의 경우에만 `ontap-nas` 지원됩니다.	`ontap-nas` `ontap-nas-economy`, 또는 `ontap-nas-flexgroup`
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF

매개 변수	설명	기본값
마나멘타LIF	클러스터 또는 SVM 관리 LIF의 IP 주소 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident가 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 과 같이 대괄호로 정의해야 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 합니다. 원활한 MetroCluster 전환은 를 MetroCluster 예참조하십시오.	"10.0.0.1"," [2001:1234:ABCD::fefe]"
다타LIF	프로토콜 LIF의 IP 주소입니다. NetApp에서는 을 지정할 것을 `dataLIF` 권장합니다. 제공되지 않는 경우 Trident는 SVM에서 데이터 LIF를 가져옵니다. NFS 마운트 작업에 사용할 FQDN(정규화된 도메인 이름)을 지정할 수 있습니다. 이렇게 하면 라운드 로빈 DNS를 생성하여 여러 데이터 LIF의 로드 밸런싱을 수행할 수 있습니다. 초기 설정 후에 변경할 수 있습니다. 을 참조하십시오. IPv6 플래그를 사용하여 Trident가 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 과 같이 대괄호로 정의해야 `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 합니다. * MetroCluster의 경우 생략합니다. * 를 MetroCluster 예참조하십시오.	지정되지 않은 경우 SVM에서 지정 주소 또는 파생(권장하지 않음)
'VM'입니다	사용할 스토리지 가상 머신입니다 *MetroCluster의 경우 생략합니다. * 를 참조하십시오 MetroCluster 예.	SVM 'managementLIF'가 지정된 경우에 파생됩니다
자동 내보내기 정책	자동 엑스포트 정책 생성 및 업데이트 [Boolean] 활성화 Trident는 및 `autoExportCIDRs` 옵션을 사용하여 `autoExportPolicy` 엑스포트 정책을 자동으로 관리할 수 있습니다.	거짓
자동 내보내기	이 설정된 경우에 대해 Kubernetes 노드 IP를 필터링하는 CIDR `autoExportPolicy` 목록입니다. Trident는 및 `autoExportCIDRs` 옵션을 사용하여 `autoExportPolicy` 엑스포트 정책을 자동으로 관리할 수 있습니다.	["0.0.0.0/0",":/0"]'
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""
'고객증명서'	Base64 - 클라이언트 인증서의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
'clientPrivateKey' 입니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
신탁인증서다	Base64 - 신뢰할 수 있는 CA 인증서의 인코딩된 값입니다. 선택 사항. 인증서 기반 인증에 사용됩니다	""
'사용자 이름'	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	

매개 변수	설명	기본값
"암호"	클러스터/SVM에 연결하기 위한 비밀번호입니다. 자격 증명 기반 인증에 사용됩니다. Active Directory 인증에 대해서는 다음을 참조하세요. "Active Directory 자격 증명을 사용하여 백엔드 SVM에 Trident 인증".	
'토르agePrefix'	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사 설정한 후에는 업데이트할 수 없습니다 참고 ontap-nas-economy를 사용하고 storagePrefix가 24자 이상인 경우, qtree에는 스토리지 접두사가 포함되지 않지만 볼륨 이름에는 포함됩니다.	"트리덴트"
골재	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. 참고 SVM에서 애그리게이트가 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident에 자동으로 업데이트됩니다. Trident에서 볼륨을 프로비저닝하도록 특정 애그리게이트를 구성한 경우, 해당 애그리게이트의 이름이 변경되거나 SVM에서 제거되면 SVM 애그리게이트를 폴링하는 동안 백엔드가 Trident에서 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 SVM에 있는 애그리게이트로 변경하거나 해당 애그리게이트를 완전히 제거해야 합니다. **AFX** 저장 시스템에 대해서는 지정하지 마세요**.** • `useBalancedPlacement`이(가) `true`(으)로 설정된 경우 지정하지 마십시오.*	""
`useBalancedPlacement`	새 볼륨에 대한 애그리게이트 선택을 ONTAP에 위임합니다[Boolean]. ONTAP 9.19.1 이상이 필요하며 `useREST`을(를) `true`로 설정해야 합니다. `aggregate`와 함께 사용할 수 없습니다. 요구 사항 및 예시는 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오. 명시된 경우**, ONTAP 9.19.1** 이상 버전의 **AFX** 스토리지 시스템에서는 항상 **`true`**으로 설정해야 합니다**.**	`false`; `true` ONTAP 9.19.1 이상의 AFX 스토리지 시스템의 경우
제한선태사용법	사용량이 이 백분율을 초과하면 프로비저닝에 실패합니다. * Amazon FSx for ONTAP 에는 적용되지 않습니다*. **AFX** 저장 시스템에 대해서는 지정하지 마세요**.**	""(기본적으로 적용되지 않음)

매개 변수	설명	기본값
flexgroupAggregateList 를 참조하십시오	프로비저닝을 위한 애그리게이트 목록(선택 사항, SVM에 할당되어야 함) SVM에 할당된 모든 애그리게이트는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다. ONTAP-NAS-FlexGroup * 스토리지 드라이버에 대해 지원됩니다. 참고 SVM에서 집계 목록이 업데이트되면 Trident 컨트롤러를 재시작하지 않고도 SVM을 폴링하여 Trident의 목록이 자동으로 업데이트됩니다. 볼륨을 프로비저닝하기 위해 Trident에서 특정 집계 목록을 구성한 경우, 해당 집계 목록의 이름이 변경되거나 SVM에서 이동되면 SVM 집계를 폴링하는 동안 Trident에서 백엔드가 실패 상태로 전환됩니다. 백엔드를 다시 온라인 상태로 전환하려면 집계 목록을 SVM에 있는 다른 목록으로 변경하거나 완전히 제거해야 합니다.	""
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 크면 프로비저닝에 실패합니다.	""(기본적으로 적용되지 않음)
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true} 사용하지 마십시오 `debugTraceFlags` 문제 해결 및 자세한 로그 덤프가 필요한 경우를 제외하고	null입니다
`nasType`	NFS 또는 SMB 볼륨 생성을 구성합니다. 옵션은 다음과 같습니다 `nfs` , `smb` 또는 null. null로 설정하면 기본적으로 NFS 볼륨이 사용됩니다. *지정된 경우 항상 다음으로 설정됩니다. `nfs` AFX 저장 시스템*용.	`nfs`
nfsMountOptions를 선택합니다	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 사용하도록 대체됩니다. 스토리지 클래스 또는 구성 파일에 마운트 옵션이 지정되지 않은 경우 Trident는 연결된 영구 볼륨에 마운트 옵션을 설정하지 않습니다.	""
"케트리스퍼플랙스볼륨"	FlexVol당 최대 qtree, 범위 [50, 300]에 있어야 함	"200"
`smbShare`	Microsoft 관리 콘솔 또는 ONTAP CLI를 사용하여 생성된 SMB 공유의 이름, Trident에서 SMB 공유를 생성할 수 있는 이름, 볼륨에 대한 일반적인 공유 액세스를 방지하기 위해 매개 변수를 비워 둘 수 있습니다. 이 매개 변수는 사내 ONTAP의 경우 선택 사항입니다. 이 매개변수는 ONTAP 백엔드에 대한 아마존 FSx에 필요하며 비워둘 수 없습니다.	`smb-share`

매개 변수	설명	기본값
'useREST'	ONTAP REST API를 사용하기 위한 부울 매개변수입니다. `useREST`설정 시 `true` , Trident 백엔드와 통신하기 위해 ONTAP REST API를 사용합니다. `false` Trident 백엔드와 통신하기 위해 ONTAPI(ZAPI) 호출을 사용합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한, 사용되는 ONTAP 로그인 역할에는 다음에 대한 액세스 권한이 있어야 합니다. `ontapi` 애플리케이션. 이는 사전 정의된 것에 의해 충족됩니다. `vsadmin` 그리고 `cluster-admin` 역할. Trident 24.06 릴리스 및 ONTAP 9.15.1 이상부터 `useREST` 로 설정됩니다 `true` 기본적으로; 변경 `useREST` 에게 `false` ONTAPI(ZAPI) 호출을 사용합니다. *지정된 경우 항상 다음으로 설정됩니다. `true` AFX 저장 시스템*용.	`true` ONTAP 9.15.1 이상, 그렇지 않은 경우 `false`.
`limitVolumePoolSize`	ONTAP-NAS-이코노미 백엔드에서 Qtree를 사용할 때 가장 필요한 최대 FlexVol 크기입니다.	""(기본적으로 적용되지 않음)
`denyNewVolumePools`	못하도록백 엔드가 새 FlexVol 볼륨을 생성하지 포함하도록 `ontap-nas-economy` 해당 qtree를 제한합니다. 기존 FlexVol만 새 PVS 프로비저닝에 사용됩니다.	
`adAdminUser`	SMB 공유에 대한 전체 액세스 권한을 가진 Active Directory 관리자 사용자 또는 사용자 그룹입니다. 이 매개변수를 사용하여 SMB 공유에 대한 모든 권한을 가진 관리자 권한을 부여할 수 있습니다.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다 `defaults` 섹션을 참조하십시오. 예를 들어, 아래 구성 예제를 참조하십시오.

매개 변수	설명	기본값
'팩시배부	Qtree에 공간 할당	"참"
'예비공간'	공간 예약 모드, "없음"(씬) 또는 "볼륨"(일반)	"없음"
냅샷정책	사용할 스냅샷 정책입니다	"없음"
"qosPolicy"	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다	""
적응성 QosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀/백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다. ONTAP에서 지원되지 않음 - NAS - 이코노미	""
안산예비역	스냅숏용으로 예약된 볼륨의 백분율입니다	"0"인 경우 `snapshotPolicy` "없음"이고, 그렇지 않으면""입니다.
'plitOnClone’을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	"거짓"

매개 변수	설명	기본값
암호화	NetApp 볼륨 암호화(NVE)를 새 볼륨에서 활성화합니다. 기본값은 `false`입니다. 이 옵션을 사용하려면 클러스터에서 NVE 라이선스가 있고 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 활성화됩니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방법".	"거짓"
'계층화 정책'	"없음"을 사용하는 계층화 정책	
유니크권한	모드를 선택합니다	NFS 볼륨의 경우 "777", SMB 볼륨의 경우 비어 있음(해당 없음
나프산디렉토리	에 액세스를 제어합니다 `.snapshot` 디렉토리	`true`, `false` (명시적으로 설정).
엑포트정책	사용할 엑스포트 정책	"기본값"
'생태성 스타일'을 참조하십시오	새로운 볼륨에 대한 보안 스타일 NFS를 지원합니다 `mixed` 및 `unix` 보안 스타일. SMB 지원 `mixed` 및 `ntfs` 보안 스타일.	NFS 기본값은 입니다 `unix`. SMB 기본값은 입니다 `ntfs`.
`nameTemplate`	사용자 지정 볼륨 이름을 생성하는 템플릿입니다.	""

참고 Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9 .8 이상이 필요합니다. 비공유 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 제한을 적용합니다.

볼륨 프로비저닝의 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"
```

을 위한 `ontap-nas` 그리고 `ontap-nas-flexgroups` Trident 이제 FlexVol snapshotReserve 백분율과 PVC에 맞게 올바르게 조정되도록 새로운 계산을 사용합니다. 사용자가 PVC를 요청하면 Trident 새로운 계산을 사용하여 더 많은 공간을 가진 원래 FlexVol 생성합니다. 이 계산은 사용자가 PVC에서 요청한 쓰기 가능 공간을 받고, 요청한 것보다 적은 공간을 받지 않도록 보장합니다. v21.07 이전에는 사용자가 스냅샷 예약 비율을 50%로 설정한 PVC(예: 5GiB)를 요청하면 2.5GiB의 쓰기 가능 공간만 확보되었습니다. 이는 사용자가 요청한 것이 전체 볼륨이기 때문입니다. `snapshotReserve` 그 중 일부입니다. Trident 21.07을 사용하면 사용자가 요청하는 것은 쓰기 가능한 공간이며 Trident 이를 정의합니다. `snapshotReserve` 전체 볼륨에 대한 백분율로 나타낸 숫자입니다. 이것은 적용되지 않습니다 `ontap-nas-economy` . 작동 방식을 알아보려면 다음 예를 참조하세요.

계산은 다음과 같습니다.

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

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

```
Vserver   Volume       Aggregate    State      Type       Size  Available Used%
--------- ------------ ------------ ---------- ---- ---------- ---------- -----
          _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4
                       online     RW         10GB     5.00GB    0%
          _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba
                       online     RW          1GB    511.8MB    0%
2 entries were displayed.
```

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

최소 구성의 예

다음 예에서는 대부분의 매개 변수를 기본값으로 두는 기본 구성을 보여 줍니다. 이는 백엔드를 정의하는 가장 쉬운 방법입니다.

참고	Trident가 있는 NetApp ONTAP에서 Amazon FSx를 사용하는 경우 IP 주소 대신 LIF에 대한 DNS 이름을 지정하는 것이 좋습니다.

▼ **ONTAP NAS** 경제도 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** 예

전환 및 전환 중에 백엔드 정의를 수동으로 업데이트할 필요가 없도록 백엔드를 구성할 수 있습니다 "SVM 복제 및 복구".

원활한 스위치오버 및 스위치백의 경우 를 사용하여 SVM을 지정합니다 `managementLIF` 를 생략합니다 `dataLIF` 및 `svm` 매개 변수. 예를 들면 다음과 같습니다.

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

▼ **SMB** 볼륨의 예입니다

```
---
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 인증의 예

이는 최소 백엔드 구성의 예입니다. `clientCertificate`, `clientPrivateKey`, 및 `trustedCACertificate` (신뢰할 수 있는 CA를 사용하는 경우 선택 사항)는 에 채워집니다 `backend.json` 그리고 각각 클라이언트 인증서, 개인 키 및 신뢰할 수 있는 CA 인증서의 base64로 인코딩된 값을 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 정책의 예

이 예에서는 Trident에서 동적 엑스포트 정책을 사용하여 엑스포트 정책을 자동으로 생성 및 관리하도록 하는 방법을 보여 줍니다. 및 `ontap-nas-flexgroup` 드라이버에도 동일하게 `ontap-nas-economy` 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
```

▼ 균형 잡힌 배치 예시

이 예제는 ONTAP 균형 배치 기능을 활성화하여 ONTAP이 새 볼륨에 대한 애그리게이트를 선택하도록 합니다. 균형 배치 기능을 사용하려면 ONTAP 9.19.1 이상 버전이 필요합니다. 자세한 내용은 "ONTAP 백엔드와 함께 균형 잡힌 배치 사용"을 참조하십시오.

```
---
version: 1
```

```
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
useBalancedPlacement: true
```

▼ **IPv6** 주소 예

이 예에서는 를 보여 줍니다 `managementLIF` IPv6 주소 사용.

```
---
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 **ONTAP**용 **Amazon FSx**의 예

를 클릭합니다 `smbShare` SMB 볼륨을 사용하는 ONTAP용 FSx에 매개 변수가 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_
```

▼ **nameTemplate**이 포함된 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 풀의 백엔드 예

아래 표시된 샘플 백엔드 정의 파일에서 와 같은 모든 스토리지 풀에 대한 특정 기본값이 설정됩니다 `spaceReserve` 없음, `spaceAllocation` 거짓일 경우, 및 `encryption` 거짓일 때. 가상 풀은 스토리지 섹션에 정의됩니다.

Trident는 "Comments" 필드에 프로비저닝 레이블을 설정합니다. 설명은 의 FlexVol `ontap-nas` 또는 의 FlexGroup에 `ontap-nas-flexgroup` 설정됩니다. Trident는 프로비저닝 시 가상 풀에 있는 모든 레이블을 스토리지 볼륨에 복제합니다. 편의를 위해 스토리지 관리자는 가상 풀 및 그룹 볼륨별로 레이블을 레이블별로 정의할 수 있습니다.

이 예에서는 일부 스토리지 풀이 자체적으로 설정됩니다 `spaceReserve`, `spaceAllocation`, 및 `encryption` 일부 풀은 기본값을 재정의합니다.

▼ **ONTAP NAS**의 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**의 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** 경제도 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 **StorageClasses**에 매핑합니다

다음 StorageClass 정의는 가상 풀의 백엔드 예을 참조합니다. `parameters.selector` 필드를 사용하여 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 지정합니다. 볼륨은 선택된 가상 풀에 정의된 속성을 갖습니다.

- The `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"
```

- The `protection-not-gold` StorageClass는 `ontap-nas-flexgroup`백엔드의 세 번째 및 네 번째 가상 풀에 매핑됩니다. 이 풀들만이 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"
```

- The `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"
```

- The `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"
```

- The `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 충족되는지 확인합니다.

업데이트 **dataLIF** 초기 구성 후

초기 구성 후 다음 명령을 실행하여 새 백엔드 JSON 파일에 업데이트된 데이터 LIF를 제공할 수 있습니다.

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

> 참고 PVC가 하나 또는 여러 개의 Pod에 연결된 경우, 새로운 데이터 LIF가 적용되려면 해당하는 모든 Pod를 종료한 다음 다시 불러와야 합니다.

보안 **SMB** 예시

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
```

참고 추가했는지 확인하세요 `annotations` 보안 SMB를 활성화합니다. 보안 SMB는 백엔드 또는 PVC에 설정된 구성과 관계없이 주석 없이는 작동하지 않습니다.

ontap-nas-economy 드라이버를 사용한 스토리지 클래스 예제

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
```

```
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

단일 **AD** 사용자가 있는 **PVC** 예

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

여러 **AD** 사용자가 있는 **PVC** 예

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
```

```
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

ONTAP 백엔드와 함께 균형 잡힌 배치 사용

ONTAP 백엔드에서 ONTAP 균형 배치 기능을 활성화할 수 있습니다. 균형 배치 기능을 사용하면 ONTAP이 각 새 볼륨에 대해 최적의 애그리게이트를 선택하므로 Trident에서 애그리게이트 할당을 관리할 필요가 없습니다.

균형 배치 방식의 작동 방식

기본적으로 Trident는 SVM에 할당된 애그리게이트를 검색하고, 각 애그리게이트를 스토리지 풀로 처리하며, 볼륨을 프로비저닝할 때 애그리게이트를 선택합니다.

균형 배치를 활성화하면:

- Trident는 개별 애그리게이트를 보고하는 대신 전체 SVM을 단일 스토리지 풀로 보고합니다.
- ONTAP는 새로운 볼륨마다 애그리게이트를 선택하고 사용 가능한 애그리게이트 전체에 걸쳐 볼륨 균형을 맞춥니다.
- Trident는 ONTAP `/application/containers` REST API를 사용하여 볼륨을 생성합니다.

통합 ONTAP 시스템(AFF, FAS 및 ASA)과 NetApp AFX 스토리지 시스템에 대해 균형 배치가 가능합니다.

요구 사항

- 백엔드는 ONTAP 9.19.1 이상 버전을 사용해야 합니다.
- Trident는 REST를 사용하여 ONTAP과 통신해야 합니다. ZAPI에서는 균형 배치를 사용할 수 없습니다.
- 백엔드 구성에서 애그리게이트를 지정해서는 안 됩니다. 균형 배치와 명시적인 애그리게이트 선택을 함께 사용할 수 없습니다.

기본값

`useBalancedPlacement`의 기본값은 스토리지 시스템에 따라 다릅니다.

스토리지 시스템	기본값	참고
통합 ONTAP(AFF, FAS, ASA)	거짓입니다	균형 배치를 활성화하지 않으면 Trident는 애그리게이트를 선택합니다.
NetApp AFX (ONTAP 9.19.1 이상)	"참"입니다	균형 잡힌 배치가 필요합니다. `useBalancedPlacement`을 `false`로 설정하면 백엔드 유효성 검사가 실패합니다.
NetApp AFX (ONTAP 9.19.1 이전 버전)	거짓입니다	균형 잡힌 배치는 불가능합니다.

구성 옵션

매개 변수	설명	기본값
`useBalancedPlacement`	새 볼륨에 대한 애그리게이트 선택을 ONTAP에 위임합니다. 이 옵션은 백엔드 구성의 루트 또는 개별 가상 풀에서 설정합니다.	`false`; `true` ONTAP 9.19.1 이상의 AFX용

가상 풀을 활용한 균형 잡힌 배치

균형 잡힌 배치 방식은 가상 풀의 유무와 관계없이 작동합니다.

- 가상 풀이 없으면 Trident는 SVM을 단일 스토리지 풀로 보고하고 균형 배치를 사용하여 모든 새 볼륨을 생성합니다.
- 가상 풀을 사용하는 경우 Trident는 평소와 같이 가상 풀을 보고하고 볼륨을 해당 가상 풀과 연결합니다. 연결된 가상 풀에 균형 배치가 활성화된 경우 ONTAP는 해당 볼륨에 대한 애그리게이트를 선택합니다.

Set `useBalancedPlacement` 모든 가상 풀에 적용하려면 백엔드 구성의 최상위 레벨에 설정하고, 선택적으로 활성화하려면 개별 가상 풀에 설정하십시오.

예

다음 예시는 통합 ONTAP 백엔드에서 균형 잡힌 배치를 가능하게 합니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: cluster-admin
password: <password>
useBalancedPlacement: true
```

다음 예시는 `TridentBackendConfig`을 사용하여 균형 잡힌 배치를 활성화합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: nas-balanced-placement
  managementLIF: 10.0.0.1
  svm: trident_svm
  useBalancedPlacement: true
  credentials:
    name: backend-tbc-ontap-nas-secret
```

다음 예시는 `gold` 가상 풀에서만 균형 배치를 활성화합니다. Trident는 `bronze` 가상 풀과 일치하는 볼륨에 대해 계속해서 애그리게이트를 선택합니다.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-virtual-pools
managementLIF: 10.0.0.1
svm: trident_svm
username: cluster-admin
password: <password>
storage:
- labels:
    performance: gold
  useBalancedPlacement: true
- labels:
    performance: bronze
```

백엔드 유효성 검사 오류

다음과 같은 경우 백엔드 생성 또는 업데이트가 실패합니다.

구성	결과
`useBalancedPlacement: true` ONTAP 9.19.1 이전 버전	유효성 검사에 실패했습니다. ONTAP 9.19.1 이상 버전으로 업그레이드하십시오.
`useBalancedPlacement: true` ZAPI를 사용하는 백엔드에서	유효성 검사에 실패했습니다. 백엔드를 REST를 사용하도록 구성하십시오.

구성	결과
`useBalancedPlacement: true` 특정 애그리게이트와 결합	유효성 검사에 실패했습니다. 애그리게이트 구성을 제거하거나 균형 배치 기능을 비활성화하십시오.
`useBalancedPlacement: false` ONTAP 9.19.1 이상 버전이 설치된 AFX에서 사용 가능	유효성 검사에 실패했습니다. ONTAP 9.19.1 이상 버전에서 AFX를 사용하려면 균형 배치가 필요합니다.

NetApp ONTAP용 Amazon FSx

Trident를 **Amazon FSx for NetApp ONTAP**와 함께 사용해 보십시오

"NetApp ONTAP용 Amazon FSx"는 NetApp ONTAP 스토리지 운영 체제를 기반으로 하는 파일 시스템을 실행하는 완전 관리형 AWS 서비스입니다. AWS의 확장성과 운영 간편성과 함께 ONTAP의 기능, 성능 및 관리를 제공합니다. 파일 시스템은 Amazon FSx의 기본 리소스이며 온프레미스 ONTAP 클러스터와 유사합니다. 각 파일 시스템에는 하나 이상의 스토리지 가상 머신(SVM)이 포함되고, 각 SVM에는 파일과 디렉터리를 저장하는 하나 이상의 볼륨이 포함됩니다. 이 통합을 통해 Amazon Elastic Kubernetes Service(EKS)에서 실행되는 Kubernetes 클러스터는 블록 및 파일 워크로드를 위해 ONTAP 기반 영구 볼륨을 프로비저닝할 수 있습니다.

요구 사항

"Trident 요구사항"FSx for ONTAP를 Trident와 통합하려면 다음이 필요합니다.

- kubbtl이 설치된 기존 Amazon EKS 클러스터 또는 자체 관리 Kubernetes 클러스터
- 클러스터의 작업자 노드에서 연결할 수 있는 NetApp ONTAP 파일 시스템용 기존 Amazon FSx 및 SVM(Storage Virtual Machine).
- 에 대해 준비된 작업자 노드입니다 "NFS 또는 iSCSI".

> 참고 Amazon Linux 및 Ubuntu에 필요한 노드 준비 단계를 따라야 합니다 "Amazon Machine Images(아마존 머신 이미지)" (AMI) EKS AMI 유형에 따라 다릅니다.

고려 사항

- SMB 볼륨:
  - SMB 볼륨은 를 사용하여 지원됩니다 `ontap-nas` 드라이버만 해당.
  - Trident EKS 애드온에서는 SMB 볼륨이 지원되지 않습니다.
  - Trident는 Windows 노드에서 실행되는 Pod에 마운트된 SMB 볼륨만 지원합니다. 자세한 내용은 "SMB 볼륨 프로비저닝을 위한 준비"를 참조하십시오.
- Trident 24.02 이전에는 자동 백업이 활성화된 Amazon FSx 파일 시스템에서 생성된 볼륨을 Trident에서 삭제할 수 없었습니다. Trident 24.02 이상에서 이 문제를 방지하려면 `fsxFilesystemID` AWS FSx for ONTAP의 백엔드 구성 파일에, AWS, `apikey` AWS `apiRegion` 및 AWS를 `secretKey` 지정합니다.

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

Trident SAN/iSCSI 및 EBS-CSI 드라이버 동시 사용

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

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

인증

Trident는 두 가지 인증 모드를 제공합니다.

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

  경고 | Trident은 SVM 사용자로 실행하거나 동일한 역할을 가진 다른 이름의 사용자로 실행해야 `vsadmin` 합니다. Amazon FSx for NetApp ONTAP에는 `fsxadmin` ONTAP 클러스터 사용자를 제한적으로 대체하는 사용자가 `admin` 있습니다. Trident와 함께 를 사용하는 `vsadmin` 것이 좋습니다.

- 인증서 기반: Trident은 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 아마존 머신 이미지**(AMI)**

EKS 클러스터는 다양한 운영 체제를 지원하지만 AWS는 컨테이너 및 EKS에 대해 특정 AMI(Amazon Machine Images)를 최적화했습니다. 다음 AMI는 NetApp Trident 25.02에서 테스트되었습니다.

아미	NAS	NAS - 경제형	iSCSI	iSCSI 경제
AL2023_x86_64_STANDARD를 참조하십시오	예	예	예	예
AL2_x86_64를 참조하십시오	예	예	예 *	예 *

BOTTLEROCKET_x86_64를 참조하십시오	예**	예	해당 없음	해당 없음
AL2023_ARM_64_STANDARD를 참조하십시오	예	예	예	예
AL2_ARM_64를 참조하십시오	예	예	예 *	예 *
BOTTLEROCKET_ARM_64를 참조하십시오	예**	예	해당 없음	해당 없음

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

참고	원하는 AMI가 여기에 나열되지 않은 경우 지원되지 않는다는 의미는 아닙니다. 단순히 테스트를 거치지 않았음을 의미합니다. 이 목록은 AMI가 작동하는 방법에 대한 가이드 역할을 합니다.

- 다음을 사용하여 수행한 테스트:
- EKS 버전: 1.32
- 설치 방법: Helm 25.06 및 AWS 추가 기능 25.06
- NAS의 경우 NFSv3과 NFSv4.1이 모두 테스트되었습니다.
- SAN 전용 iSCSI는 테스트되었으며 NVMe-oF는 테스트되지 않았습니다.
- 수행된 테스트 *:
- 생성 : 저장 클래스, PVC, POD
- 삭제: Pod, PVC(일반, qtree/LUN – 경제성, NAS와 AWS 백업)

자세한 내용을 확인하십시오

- "NetApp ONTAP용 Amazon FSx 문서"
- "NetApp ONTAP용 Amazon FSx 블로그 게시물"

IAM 역할 및 **AWS Secret**을 생성합니다

AWS 자격 증명을 명시적으로 제공하는 대신 AWS IAM 역할로 인증하여 Kubernetes Pod를 구성하여 AWS 리소스에 액세스할 수 있습니다.

참고	AWS IAM 역할을 사용하여 인증하려면 EKS를 사용하여 Kubernetes 클러스터를 구축해야 합니다.

AWS Secrets Manager 암호를 생성합니다

Trident는 사용자를 위해 스토리지 관리를 위해 FSx 가상 서버에 대해 API를 발행하므로 이를 위해 자격 증명이 필요합니다. 이러한 자격 증명을 전달하는 안전한 방법은 AWS Secrets Manager 암호를 사용하는 것입니다. 따라서 아직 계정이 없는 경우 vsadmin 계정의 자격 증명이 포함된 AWS Secrets Manager 암호를 생성해야 합니다.

이 예에서는 Trident CSI 자격 증명을 저장하기 위한 AWS Secrets Manager 암호를 생성합니다.

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

IAM 정책을 생성합니다

Trident를 올바르게 실행하려면 AWS 권한도 필요합니다. 따라서 Trident에 필요한 사용 권한을 부여하는 정책을 만들어야 합니다.

다음 예에서는 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 ID** 또는 **IAM** 역할 생성

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

포드 아이덴티티

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

EKS 클러스터에 **Pod Identity** 설치:

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

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

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

trust-relationship.json을 생성합니다:

EKS 서비스 주체가 Pod Identity에 대한 이 역할을 수행할 수 있도록 trust-relationship.json 파일을 생성하세요. 그런 다음 다음 신뢰 정책을 사용하여 역할을 생성하세요.

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

trust-relationship.json 파일:

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

```
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

IAM 역할에 역할 정책 첨부:

이전 단계의 역할 정책을 생성된 IAM 역할에 연결합니다.

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

포드 **ID** 연결 생성:

IAM 역할과 Trident 서비스 계정(trident-controller) 간에 Pod ID 연결을 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 연결**(IRSA)**을 위한 **IAM** 역할

AWS CLI 사용:

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

- trust-relationship.json 파일: *

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
```

```
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

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

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

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

- IAM 정책에 IAM 역할 연결 *:

역할이 생성되면 다음 명령을 사용하여 정책(위 단계에서 만든 정책)을 역할에 연결합니다.

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

- OICD 공급자가 연결되었는지 확인 *:

OIDC 공급자가 클러스터와 연결되어 있는지 확인합니다. 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 연결합니다.

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

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

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
```

```
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Trident를 설치합니다

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.

- 헬름
- EKS 추가 기능

스냅샷 기능을 사용하려면 CSI 스냅샷 컨트롤러 애드온을 설치하십시오. 자세한 내용은 을 "CSI 볼륨에 대해 스냅샷 기능을 활성화합니다" 참조하십시오.

Helm을 통해 Trident를 설치합니다

포드 아이덴티티

1. Trident Helm 저장소 추가:

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

2. 다음 예를 사용하여 Trident를 설치하세요.

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

명령을 사용하여 이름, 네임스페이스, 차트, 상태, 앱 버전 및 수정 번호와 같은 설치 세부 정보를 검토할 수 `helm list` 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 저장소 추가:

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

2. 클라우드 공급자 및 *클라우드 ID*에 대한 값을 설정합니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

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

참고

iSCSI를 사용하려면 클라이언트 머신에서 iSCSI가 활성화되어 있는지 확인하세요. AL2023 Worker 노드 OS를 사용하는 경우, helm 설치 시 node prep 매개변수를 추가하여 iSCSI 클라이언트 설치를 자동화할 수 있습니다.

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

EKS 애드온을 통해 **Trident**를 설치합니다

Trident EKS 애드온에는 최신 보안 패치 및 버그 수정이 포함되어 있으며 AWS에서 Amazon EKS와 함께 사용할 수 있다는 것이 검증되었습니다. EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장하고

애드온을 설치, 구성 및 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

필수 구성 요소

AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항이 있는지 확인하십시오.

- 애드온 가입이 있는 Amazon EKS 클러스터 계정입니다
- AWS 마켓플레이스에 대한 AWS 권한:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 유형: Amazon Linux 2 (AL2_x86_64) 또는 Amazon Linux 2 Arm (AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

AWS에 대해 **Trident** 애드온을 활성화합니다

관리 콘솔과 직접 연결되어 있습니다

1. 에서 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 선택합니다.
4. Add-ons * 를 선택한 다음 * Get more add-ons * 를 선택합니다.
5. 추가 기능을 선택하려면 다음 단계를 따르세요.
   a. **AWS Marketplace** 추가 기능 섹션까지 아래로 스크롤하여 검색 상자에 *"Trident"*를 입력합니다.
   b. Trident by NetApp 상자의 오른쪽 상단에 있는 확인란을 선택하세요.
   c. 다음 * 을 선택합니다.
6. 선택한 추가 기능 구성 * 설정 페이지에서 다음을 수행합니다.

   참고 **Pod Identity** 연결을 사용하는 경우 이 단계를 건너뜁니다.

   a. 사용할 * 버전 * 을 선택합니다.
   b. IRSA 인증을 사용하는 경우 선택적 구성 설정에서 사용 가능한 구성 값을 설정해야 합니다.
      - 사용할 * 버전 * 을 선택합니다.
      - 추가 기능 구성 스키마*를 따르고 *구성 값 섹션의 **configurationValues** 매개변수를 이전 단계에서 만든 role-arn으로 설정합니다(값은 다음 형식이어야 함).

```
{

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

```
}
```

+
충돌 해결 방법으로 재정의 를 선택한 경우 기존 애드온에 대한 하나 이상의 설정을 Amazon EKS 애드온 설정으로 덮어쓸 수 있습니다. 이 옵션을 사용하지 않고 기존 설정과 충돌하는 경우 작업이 실패합니다. 결과 오류 메시지를 사용하여 충돌 문제를 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 자체 관리해야 하는 설정을 관리하지 않는지 확인하십시오.

7. 다음 * 을 선택합니다.
8. 검토 및 추가 * 페이지에서 * 만들기 * 를 선택합니다.

   추가 기능 설치가 완료되면 설치된 추가 기능이 표시됩니다.

AWS CLI를 참조하십시오

- 1. 생성하다 `add-on.json` 파일*:

Pod Identity의 경우 다음 형식을 사용하세요:

참고	다음을 사용하세요

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

IRSA 인증의 경우 다음 형식을 사용하세요:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

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

- 2. Trident EKS 애드온을 설치하세요.*

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

eksctl입니다

다음 명령 예에서는 Trident EKS 추가 기능을 설치합니다.

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Trident EKS 추가 기능을 업데이트합니다

관리 콘솔과 직접 연결되어 있습니다

1. Amazon EKS 콘솔을 https://console.aws.amazon.com/eks/home#/clusters엽니다.
2. 왼쪽 탐색 창에서 * 클러스터 * 를 선택합니다.
3. NetApp Trident CSI 애드온을 업데이트할 클러스터의 이름을 선택합니다.
4. Add-ons * 탭을 선택합니다.
5. Trident by NetApp * 를 선택한 다음 * 편집 * 을 선택합니다.
6. Trident by NetApp * 구성 페이지에서 다음을 수행합니다.
   a. 사용할 * 버전 * 을 선택합니다.
   b. 선택적 구성 설정 * 을 확장하고 필요에 따라 수정합니다.
   c. 변경 내용 저장 * 을 선택합니다.

AWS CLI를 참조하십시오

다음 예에서는 EKS 추가 기능을 업데이트합니다.

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl입니다

- FSxN Trident CSI 추가 기능의 현재 버전을 확인합니다. 클러스터 이름으로 교체합니다 `my-cluster`.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

- 출력 예: *

```
NAME                            VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 이전 단계의 출력에서 사용할 수 있는 업데이트 아래에 반환된 버전으로 추가 기능을 업데이트합니다.

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

옵션을 제거하고 Amazon EKS 추가 기능 설정이 기존 설정과 충돌하는 경우 `--force` Amazon EKS 추가 기능 업데이트가 실패하고 충돌 문제를 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 지정하기 전에 Amazon EKS 애드온이 관리해야 하는 설정을 관리하지 않는지 확인하십시오. 이러한 설정은 이 옵션으로 덮어써지기 때문입니다. 이 설정의 다른 옵션에 대한 자세한 내용은 을 참조하십시오 "추가 기능". Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 현장 관리".

Trident EKS 추가 기능을 제거/제거합니다

Amazon EKS 애드온을 제거하는 두 가지 옵션이 있습니다.

- * 클러스터에 애드온 소프트웨어 유지 * – 이 옵션은 모든 설정의 Amazon EKS 관리를 제거합니다. 또한 업데이트를 시작한 후 Amazon EKS에서 업데이트를 알리고 Amazon EKS 애드온을 자동으로 업데이트하는 기능도 제거합니다. 하지만 클러스터에 애드온 소프트웨어가 보존됩니다. 이 옵션을 사용하면 Amazon EKS 애드온이 아닌 자가 관리형 설치가 됩니다. 이 옵션을 사용하면 애드온에 대한 다운타임이 없습니다. `--preserve`명령의 옵션을 유지하여 추가 기능을 유지합니다.
- * 클러스터에서 애드온 소프트웨어 완전히 제거 * – NetApp는 클러스터에 종속된 리소스가 없는 경우에만 클러스터에서 Amazon EKS 애드온을 제거할 것을 권장합니다. `--preserve`추가 기능을 제거하려면 명령에서 옵션을 `delete` 제거하십시오.

참고 | 애드온에 IAM 계정이 연결되어 있으면 IAM 계정이 제거되지 않습니다.

관리 콘솔과 직접 연결되어 있습니다

1. 에서 Amazon EKS 콘솔을 엽니다 https://console.aws.amazon.com/eks/home#/clusters.
2. 왼쪽 탐색 창에서 * 클러스터 * 를 선택합니다.
3. NetApp Trident CSI 추가 기능을 제거할 클러스터의 이름을 선택합니다.
4. 추가 기능 * 탭을 선택한 다음 * Trident by NetApp *. * 를 선택합니다
5. 제거 * 를 선택합니다.
6. Remove netapp_trident-operator confirmation * 대화 상자에서 다음을 수행합니다.
   a. Amazon EKS가 애드온에 대한 설정 관리를 중지하도록 하려면 * 클러스터에서 유지 * 를 선택합니다. 추가 기능의 모든 설정을 직접 관리할 수 있도록 클러스터에 추가 소프트웨어를 유지하려는 경우 이 작업을 수행합니다.

b. netapp_trident-operator * 를 입력합니다.

c. 제거 * 를 선택합니다.

AWS CLI를 참조하십시오

클러스터 이름으로 바꾸고 `my-cluster` 다음 명령을 실행합니다.

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

eksctl입니다

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

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

스토리지 클래스 구성

이 "Kubernetes StorageClass 객체" 설정은 프로비저너를 식별하고 프로비저너에게 볼륨을 프로비저닝하는 방법을 지시합니다. 이 섹션에서는 Trident를 프로비저너로 지정하는 Kubernetes StorageClass 객체를 구성하는 방법을 보여줍니다.

StorageClass 객체를 생성합니다

FSx for ONTAP용 StorageClass를 생성하면 Trident가 백엔드 구성을 자동으로 생성합니다.

참고	스토리지 백엔드를 수동으로 구성하려면 Trident 백엔드와 스토리지 클래스를 별도로 생성하는 방법에 대한 자동 백엔드 구성 없이 Kubernetes StorageClass 생성 섹션을 참조하십시오.

필수 **StorageClass** 매개변수를 지정하세요

StorageClass를 생성할 때 다음 세 가지 매개 변수를 정의해야 합니다.

매개 변수	필수 요소입니다	유형	설명
`fsxFilesystemID`	예	문자열	FSx for NetApp ONTAP 파일 시스템 ID
'torageDriverName'입니다	예	문자열	Trident 스토리지 드라이버(예: `ontap-nas` 또는 `ontap-san`)
`credentialsName`	예	문자열	ONTAP 자격 증명용 FSx가 포함된 Kubernetes Secret의 이름

선택적 매개변수 지정

StorageClass를 통해 선택적 백엔드 매개 변수를 전달할 수 있습니다. StorageClass `parameters` 섹션에서 모든 선택적 값을 문자열로 정의하십시오. 백엔드 매개변수의 전체 목록은 다음을 참조하십시오: "FSx for NetApp ONTAP

백엔드 구성".

StorageClass 구성 파일의 예입니다.

다음 예는 자동 백엔드 구성을 트리거하는 StorageClass를 보여줍니다.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON을 참조하십시오

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

StorageClass 생성

구성 파일을 생성한 후 다음 명령을 실행하여 스토리지 클래스를 생성하십시오.

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

이제 Kubernetes와 Trident 모두에서 * BASIC-CSI * 스토리지 클래스가 표시되고 Trident는 백엔드에서 풀을 검색해야 합니다.

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

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

StorageClass를 적용하면 Trident가 백엔드를 자동으로 생성합니다. 그런 다음, 이 StorageClass를 참조하는 PersistentVolumeClaims를 생성할 수 있습니다.

백엔드 구성 상태를 확인합니다

Trident는 백엔드 생성 결과를 StorageClass 어노테이션에 기록합니다.

주석	설명
`trident.netapp.io/configuratorStatus`	구성 결과 (`Success` 또는 `Failure`)
`trident.netapp.io/configuratorMessage`	자세한 상태 또는 오류 메시지
`trident.netapp.io/configuratorName`	내부 구성자 리소스의 이름
`trident.netapp.io/managed`	StorageClass가 Trident에 의해 관리됨을 나타냅니다
`trident.netapp.io/additionalStoragePools`	이 백엔드용으로 생성된 스토리지 풀

상태를 확인하려면 다음을 실행합니다.

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

`trident.netapp.io/configuratorStatus`이(가) `Success`(으)로 설정되어 있는지 확인하십시오. 값이 `Failure`인 경우 오류에 대해 `trident.netapp.io/configuratorMessage`을(를) 검토하십시오.

추가 **FSxN** 파일 시스템 추가

동일한 StorageClass를 계속 사용하면서 추가 스토리지 용량이 필요한 경우 추가 FSxN 파일 시스템 ID를 추가하십시오.

StorageClass를 편집하고 다음 주석을 추가하세요.

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

변경 사항을 적용하면 Trident가 백엔드 구성을 업데이트하고 StorageClass 어노테이션을 업데이트합니다.

운영상의 고려사항 및 제한사항

- 자동 백엔드 구성이 있는 StorageClass를 삭제하면 일반적으로 연결된 Trident 백엔드도 함께 삭제됩니다. 이로 인해 스토리지 연결이 중단되고 실행 중인 워크로드가 중단될 수 있습니다. 관리형 StorageClass를 삭제하기 전에 영향을 검증하십시오.
- 자동 백엔드 구성은 AWS FSx for NetApp ONTAP에서만 지원됩니다.

자동 백엔드 구성 없이 **Kubernetes StorageClass** 생성

Trident 백엔드와 StorageClass를 별도로 생성하려면 다음 단계를 따르세요.

자동 백엔드 구성 작동 방식 이해

Trident는 StorageClass 정의에서 백엔드 구성을 가져옵니다. StorageClass를 적용하면 Trident는 필수 매개변수를 검증하고 백엔드를 생성한 다음 StorageClass에 상태 주석을 추가합니다.

Trident는 VolumeSnapshotClass를 한 번만 생성합니다. Trident는 이후의 StorageClasses에 대해 동일한 VolumeSnapshotClass를 재사용합니다.

Trident 백엔드를 생성합니다

Trident 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
```

JSON을 참조하십시오

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

FSx for ONTAP 드라이버 세부 정보

다음 드라이버를 사용하여 Trident를 Amazon FSx for NetApp ONTAP와 통합할 수 있습니다.

드라이버 이름	설명
'ONTAP-SAN'	프로비저닝된 각 PV는 자체 Amazon FSx for NetApp ONTAP 볼륨 내의 LUN입니다. 블록 스토리지에 권장됩니다.
'ONTAP-NAS'	프로비저닝된 각 PV는 전체 Amazon FSx for NetApp ONTAP 볼륨입니다. NFS 및 SMB에 권장됩니다.
ONTAP-SAN-이코노미	프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 LUN 수를 가진 LUN입니다.

드라이버 이름	설명
ONTAP-NAS-이코노미	프로비저닝된 각 PV는 qtree이며, Amazon FSx for NetApp ONTAP 볼륨당 구성 가능한 수의 qtree가 할당됩니다.
'ONTAP-NAS-Flexgroup'	프로비저닝된 각 PV는 Amazon FSx for NetApp ONTAP FlexGroup 볼륨입니다.

드라이버에 대한 자세한 내용은 을 참조하십시오 "NAS 드라이버" 및 "SAN 드라이버".

백엔드를 생성합니다

구성 파일을 생성한 후 다음 명령을 실행하여 Trident 백엔드 구성(TBC)을 생성하고 유효성을 검사하십시오.

- YAML 파일에서 TBC(Trident 백엔드 구성)를 생성하고 다음 명령을 실행합니다.

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

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

- Trident 백엔드 구성(TBC)이 성공적으로 생성되었는지 확인:

```
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
```

다른 구성 옵션에 대한 자세한 내용은 아래 백엔드 고급 구성 및 예제 섹션을 참조하십시오.

자동 백엔드 구성 없이 스토리지 클래스 구성

다음은 Trident 및 FSx for ONTAP에서 사용할 스토리지 클래스 구성의 예입니다.

NFS용 스토리지 클래스

이 예제를 사용하여 NFS를 사용하는 볼륨에 대한 StorageClass를 설정할 수 있습니다(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"
```

iSCSI용 스토리지 클래스

iSCSI를 사용하여 볼륨에 대한 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"
```

NFSv3 및 **AWS Bottlerocket**을 사용하는 스토리지 클래스

AWS Bottlerocket에서 NFSv3 볼륨을 프로비저닝하려면 필요한 를 `mountOptions` 스토리지 클래스에 추가합니다.

```
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
```

Trident StorageClass 속성

이러한 매개 변수는 지정된 유형의 볼륨을 프로비저닝하는 데 사용해야 하는 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공합니다	요청하십시오	에 의해 지원됩니다
미디어 [1]	문자열	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있으며, 하이브리드는 둘 모두를 의미합니다	지정된 미디어 유형입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, solidfire-SAN
프로비저닝 유형	문자열	얇고 두껍습니다	풀은 이 프로비저닝 방법을 지원합니다	프로비저닝 방법이 지정되었습니다	Thick: All ONTAP; Thin: All ONTAP & solidfire-SAN
백엔드 유형	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀이 이 백엔드 유형에 속합니다	백엔드가 지정되었습니다	모든 드라이버
스냅샷 수	불입니다	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	온탑나스, 온탑씨, 솔리드파이어씨
복제	불입니다	참, 거짓	풀은 볼륨 클론을 지원합니다	클론이 활성화된 볼륨	온탑나스, 온탑씨, 솔리드파이어씨
암호화	불입니다	참, 거짓	풀은 암호화된 볼륨을 지원합니다	암호화가 활성화된 볼륨입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroups, ONTAP-SAN
IOPS	내부	양의 정수입니다	풀은 이 범위에서 IOPS를 보장할 수 있습니다	볼륨은 이러한 IOPS를 보장합니다	solidfire-SAN

[1]: ONTAP Select 또는 FSx for ONTAP 시스템에서 지원되지 않습니다

Trident에서 볼륨을 프로비저닝하는 방법을 제어하는 데 사용되는 및 매개 변수와 스토리지 클래스가 상호 작용하는 방법에 대한 자세한 `PersistentVolumeClaim` 내용은 을 "Kubernetes 및 Trident 오브젝트"참조하십시오.

스토리지 클래스를 생성합니다

StorageClass를 구성한 후 Kubernetes에서 생성할 수 있습니다.

단계

1. Kubernetes 오브젝트이므로 를 사용하십시오 `kubectl` Kubernetes에서 생성해야 합니다.

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

2. 이제 Kubernetes와 Trident 모두에서 * BASIC-CSI * 스토리지 클래스가 표시되고 Trident는 백엔드에서 풀을 검색해야 합니다.

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

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

SMB 볼륨 제공

`ontap-nas` 드라이버를 사용하여 SMB 볼륨을 프로비저닝할 수 있습니다. 하지만 그러려면 다음 단계를 완료해야 합니다: xref:{relative_path}worker-node-prep.html#prepare-to-provision-smb-volumes["SMB 볼륨 프로비저닝을 위한 준비"].

백엔드 고급 구성 및 예제

백엔드 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	예
'내전'		항상 1
'torageDriverName'입니다	스토리지 드라이버의 이름입니다	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy
백엔드이름	사용자 지정 이름 또는 스토리지 백엔드	드라이버 이름 + "_" + dataLIF

매개 변수	설명	예
마나멘타LIF	클러스터 또는 SVM 관리 LIF의 IP 주소 정규화된 도메인 이름(FQDN)을 지정할 수 있습니다. IPv6 플래그를 사용하여 Trident가 설치된 경우 IPv6 주소를 사용하도록 설정할 수 있습니다. IPv6 주소는 대괄호로 묶어야 합니다(예: [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]). `aws`현장에서 을 제공하는 경우 `fsxFilesystemID`, Trident이 AWS에서 SVM 정보를 검색하기 때문에 를 제공할 필요가 없습니다 `managementLIF`. `managementLIF` 따라서 SVM에서 사용자에 대한 자격 증명(예: vsadmin)을 제공해야 하며 사용자에게 역할이 있어야 합니다. `vsadmin`	"10.0.0.1"," [2001:1234:ABCD::fefe]"
다타LIF	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. **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 사용할 수 있도록 설정할 수 있습니다. IPv6 주소는 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]와 같이 대괄호로 정의해야 합니다.	
자동 내보내기 정책	자동 엑스포트 정책 생성 및 업데이트 [Boolean] 활성화 Trident는 및 `autoExportCIDRs` 옵션을 사용하여 `autoExportPolicy` 엑스포트 정책을 자동으로 관리할 수 있습니다.	거짓입니다

매개 변수	설명	예
자동 내보내기	이 설정된 경우에 대해 Kubernetes 노드 IP를 필터링하는 CIDR `autoExportPolicy` 목록입니다. Trident는 및 `autoExportCIDRs` 옵션을 사용하여 `autoExportPolicy` 엑스포트 정책을 자동으로 관리할 수 있습니다.	"["0.0.0.0/0",":/0"]"
'라벨'	볼륨에 적용할 임의의 JSON 형식 레이블 세트입니다	""
'고객증명서'	Base64 - 클라이언트 인증서의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
'clientPrivateKey’입니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다	""
신탁인증서다	Base64 - 신뢰할 수 있는 CA 인증서의 인코딩된 값입니다. 선택 사항. 인증서 기반 인증에 사용됩니다.	""
'사용자 이름'	클러스터 또는 SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다. 예: vsadmin.	
"암호"	클러스터 또는 SVM에 연결하는 암호 자격 증명 기반 인증에 사용됩니다.	
'VM’입니다	사용할 스토리지 가상 머신입니다	SVM 관리 LIF가 지정된 경우에 파생됩니다.
'토르agePrefix'	SVM에서 새 볼륨을 프로비저닝할 때 사용되는 접두사 생성 후에는 수정할 수 없습니다. 이 매개 변수를 업데이트하려면 새 백엔드를 생성해야 합니다.	`trident`
제한선태사용법	* NetApp ONTAP * 용 아마존 FSx에 대해서는 지정하지 마십시오 제공된 및 `vsadmin` 에는 `fsxadmin` 애그리게이트 사용량을 검색하고 Trident를 사용하여 제한하는 데 필요한 권한이 포함되어 있지 않습니다.	사용하지 마십시오.
LimitVolumeSize	요청된 볼륨 크기가 이 값보다 큰 경우 용량 할당에 실패합니다. 또한 qtree 및 LUN에서 관리하는 볼륨의 최대 크기를 제한하고, `qtreesPerFlexvol` 옵션을 통해 FlexVol volume당 최대 qtree 수를 사용자 지정할 수 있습니다	""(기본적으로 적용되지 않음)

매개 변수	설명	예
'오만유연한'	FlexVol volume당 최대 LUN은 [50, 200] 범위에 있어야 합니다. SAN만 해당.	"'100'"
debugTraceFlags를 선택합니다	문제 해결 시 사용할 디버그 플래그입니다. 예: {"api":false, "method":true} 사용하지 마십시오 `debugTraceFlags` 문제 해결 및 자세한 로그 덤프가 필요한 경우를 제외하고	null입니다
nfsMountOptions를 선택합니다	쉼표로 구분된 NFS 마운트 옵션 목록입니다. Kubernetes 영구 볼륨의 마운트 옵션은 일반적으로 스토리지 클래스에 지정되어 있지만, 스토리지 클래스에 마운트 옵션이 지정되지 않은 경우 Trident는 스토리지 백엔드의 구성 파일에 지정된 마운트 옵션을 사용하도록 폴백합니다. 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 입니다 `nfs`, `smb`또는 `null`입니다. * 를 로 설정해야 합니다 `smb` SMB 볼륨의 경우. * null로 설정하면 기본적으로 NFS 볼륨이 설정됩니다.	`nfs`
"케트리스퍼플랙스볼륨"	FlexVol volume당 최대 qtree, 범위 [50, 300] 내에 있어야 함	`"200"`
`smbShare`	Microsoft 관리 콘솔 또는 ONTAP CLI를 사용하여 생성된 SMB 공유의 이름 또는 Trident에서 SMB 공유를 생성하도록 허용하는 이름을 지정할 수 있습니다. 이 매개변수는 ONTAP 백엔드에 대한 Amazon FSx에 필요합니다.	`smb-share`

매개 변수	설명	예
'useREST'	ONTAP REST API를 사용하는 부울 매개 변수입니다. 로 `true` 설정하면 Trident에서 ONTAP REST API를 사용하여 백엔드와 통신합니다. 이 기능을 사용하려면 ONTAP 9.11.1 이상이 필요합니다. 또한 사용되는 ONTAP 로그인 역할에는 애플리케이션에 대한 액세스 권한이 있어야 `ontap` 합니다. 이는 미리 정의된 역할과 역할에 의해 충족됩니다. `vsadmin` `cluster-admin`	거짓입니다
`aws`	AWS FSx for ONTAP의 구성 파일에서 다음을 지정할 수 있습니다. - `fsxFilesystemID`: AWS FSx 파일 시스템의 ID를 지정합니다. - `apiRegion`: AWS API 지역 이름입니다. - `apikey`: AWS API 키입니다. - `secretKey`: AWS 비밀 키입니다.	"" "" ""
`credentials`	AWS Secrets Manager에 저장할 FSx SVM 자격 증명을 지정합니다. - `name`: SVM의 자격 증명이 포함된 비밀의 ARN(Amazon Resource Name). `type`-: 로 `awsarn`설정합니다. 자세한 내용은 을 "AWS Secrets Manager 암호를 생성합니다" 참조하십시오.	

볼륨 프로비저닝을 위한 백엔드 구성 옵션

에서 이러한 옵션을 사용하여 기본 프로비저닝을 제어할 수 있습니다 `defaults` 섹션을 참조하십시오. 예를 들어, 아래 구성 예제를 참조하십시오.

매개 변수	설명	기본값
'팩시배부	LUN에 대한 공간 할당	"참"입니다
'예비공간'	공간 예약 모드, "없음"(씬) 또는 " 볼륨"(일반)	"없음"
냅샷정책	사용할 스냅샷 정책입니다	"없음"

매개 변수	설명	기본값
"qosPolicy"	생성된 볼륨에 할당할 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다. Trident에서 QoS 정책 그룹을 사용하려면 ONTAP 9 .8 이상이 필요합니다. 비공유 QoS 정책 그룹을 사용하고 정책 그룹이 각 구성 요소에 개별적으로 적용되도록 해야 합니다. 공유 QoS 정책 그룹은 모든 워크로드의 총 처리량에 대한 제한을 적용합니다.	""
적응성 QosPolicy	생성된 볼륨에 할당할 적응형 QoS 정책 그룹입니다. 스토리지 풀 또는 백엔드에서 qosPolicy 또는 adapativeQosPolicy 중 하나를 선택합니다. ONTAP에서 지원되지 않음 - NAS - 이코노미	""
안산예비역	스냅샷 "0"에 대해 예약된 볼륨의 백분율	`snapshotPolicy` `none`있다면, `else`""
'plitOnClone'을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다	거짓입니다
암호화	새 볼륨에서 NetApp 볼륨 암호화(NVE)를 활성화하고, 기본값은 로 설정합니다. `false` 이 옵션을 사용하려면 NVE 라이센스가 클러스터에서 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 사용됩니다. 자세한 내용은 다음을 "Trident가 NVE 및 NAE와 작동하는 방법"참조하십시오.	거짓입니다
`luksEncryption`	LUKS 암호화를 활성화합니다. 다음을 참조하십시오. "LUKS(Linux Unified Key Setup) 사용". SAN 전용입니다.	""
'계층화 정책'	사용할 계층화 정책 `none`	
유니크권한	모드를 선택합니다. * SMB 볼륨의 경우 비워 둡니다. *	""
'생태성 스타일'을 참조하십시오	새로운 볼륨에 대한 보안 스타일 NFS를 지원합니다 `mixed` 및 `unix` 보안 스타일. SMB 지원 `mixed` 및 `ntfs` 보안 스타일.	NFS 기본값은 입니다 `unix`. SMB 기본값은 입니다 `ntfs`.

PVC 구성

이 섹션에는 구성된 Kubernetes StorageClass를 사용하여 PV를 요청하는 PersistentVolumeClaim(PVC)을 생성하는 방법에 대한 지침이 포함되어 있습니다. 성공하면

PV를 파드에 마운트할 수 있습니다.

PVC를 작성합니다

`https://kubernetes.io/docs/concepts/storage/persistent-volumes["_PersistentVolumeClaim_"^](PVC)`는 클러스터의 `PersistentVolume`에 대한 액세스 요청입니다. `PVC`는 특정 크기의 스토리지 또는 액세스 모드를 요청하도록 구성할 수 있습니다. 연결된 `StorageClass`를 사용하여 클러스터 관리자는 `PersistentVolume` 크기 및 액세스 모드 외에도 성능이나 서비스 수준과 같은 다양한 요소를 제어할 수 있습니다.

Trident 백엔드와 StorageClass를 생성한 후 PVC를 생성할 수 있습니다. PVC가 생성되면 해당 볼륨을 Pod에 마운트할 수 있습니다.

샘플 매니페스트

다음 예시는 기본적인 PVC 구성 옵션을 보여줍니다.

RWX 액세스 **PVC**

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

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

iSCSI를 사용한 **PVC** 예제

이 예에서는 RWO 액세스가 있는 iSCSI용 기본 PVC를 보여줍니다. 이 PVC는 StorageClass와 연결되어 있습니다. `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

```
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

PVC 생성

단계

1. PVC를 작성합니다.

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

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

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

Trident에서 볼륨을 프로비저닝하는 방법을 제어하는 데 사용되는 및 매개 변수와 스토리지 클래스가 상호 작용하는 방법에 대한 자세한 `PersistentVolumeClaim` 내용은 을 "Kubernetes 및 Trident 오브젝트"참조하십시오.

애플리케이션 배포

저장 클래스 및 PVC가 생성되면 PV를 포드에 장착할 수 있습니다. 이 섹션에는 PV를 Pod에 연결하기 위한 명령 및 구성의 예가 나와 있습니다.

샘플 애플리케이션 배포

단계

1. 볼륨을 Pod에 마운트합니다.

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

다음 예는 PVC를 포드에 부착하기 위한 기본 구성을 보여줍니다. * 기본 구성 *:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
```

```
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

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

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

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

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

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

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

EKS 클러스터에서 **Trident EKS** 애드온을 구성합니다

NetApp Trident은 Kubernetes에서 Amazon FSx for NetApp ONTAP 스토리지 관리를 간소화하여 개발자와 관리자가 애플리케이션 구축에 집중할 수 있도록 지원합니다. NetApp Trident EKS 애드온에는 최신 보안 패치 및 버그 수정이 포함되어 있으며 AWS에서 Amazon EKS와 함께 사용할 수 있다는 것이 검증되었습니다. EKS 애드온을 사용하면 Amazon EKS 클러스터의 보안과 안정성을 지속적으로 보장하고 애드온을 설치, 구성 및 업데이트하는 데 필요한 작업량을 줄일 수 있습니다.

필수 구성 요소

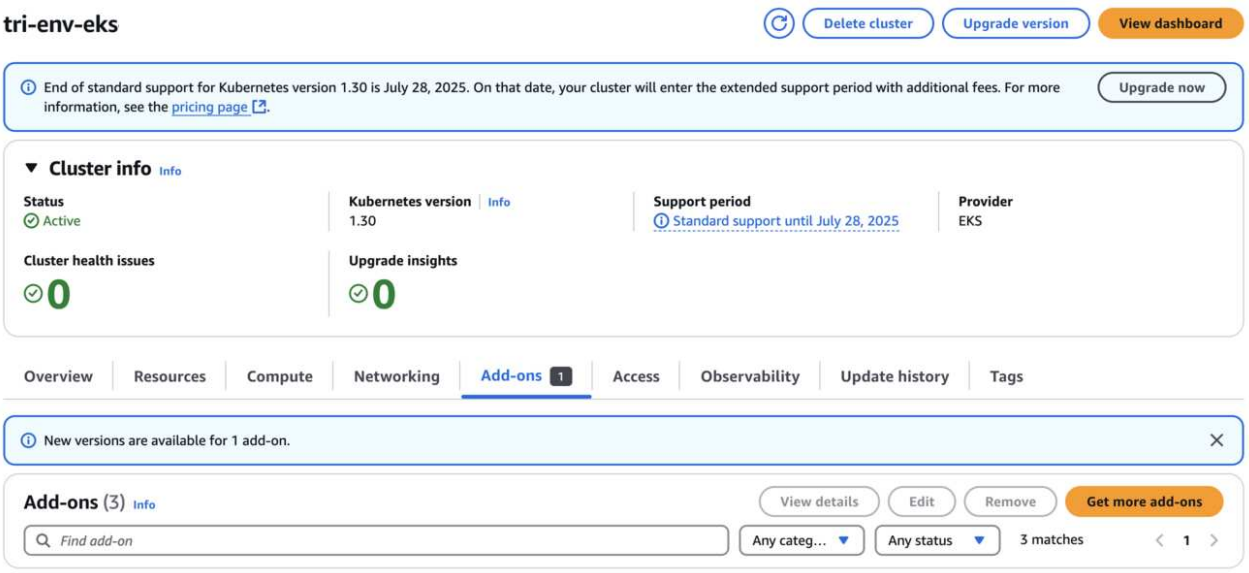
AWS EKS용 Trident 애드온을 구성하기 전에 다음 사항이 있는지 확인하십시오.

- 추가 기능을 사용할 수 있는 권한이 있는 Amazon EKS 클러스터 계정입니다. 을 "Amazon EKS 애드온 "참조하십시오.

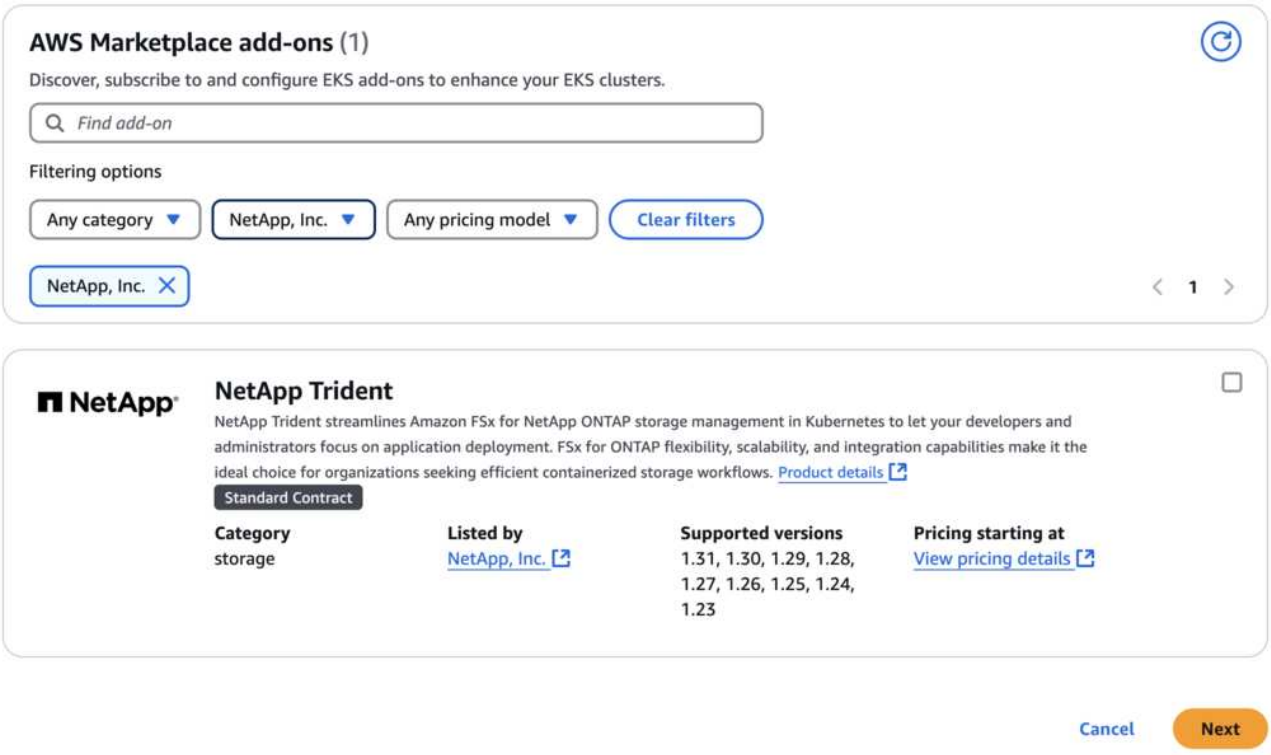
- AWS 마켓플레이스에 대한 AWS 권한:
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
- AMI 유형: Amazon Linux 2 (AL2_x86_64) 또는 Amazon Linux 2 Arm (AL2_ARM_64)
- 노드 유형: AMD 또는 ARM
- 기존 Amazon FSx for NetApp ONTAP 파일 시스템

단계

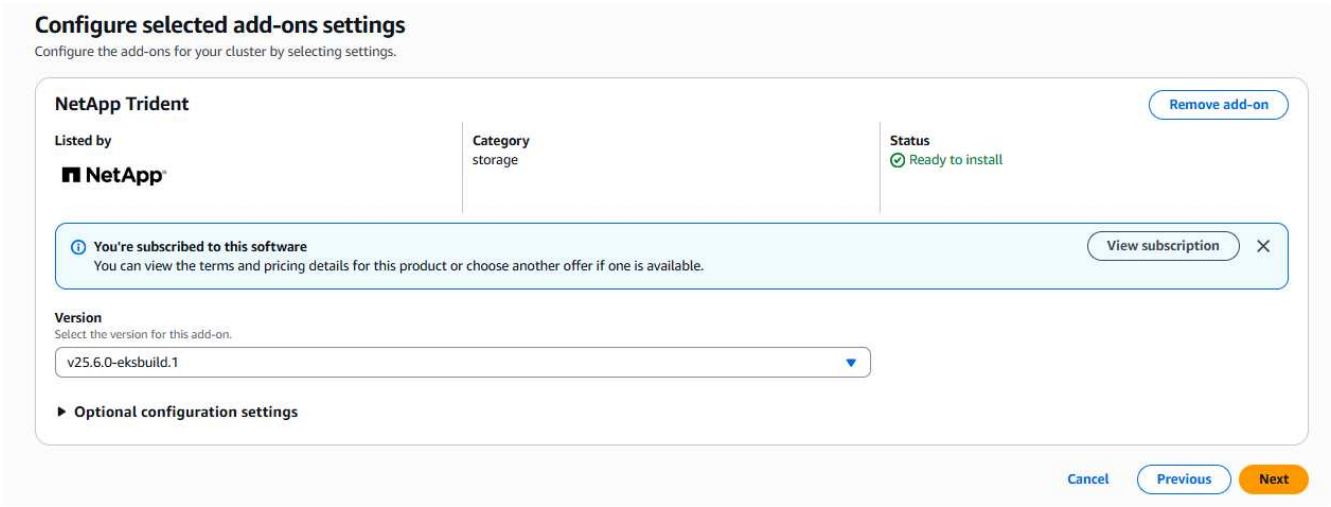
1. EKS Pod에서 AWS 리소스에 액세스할 수 있도록 IAM 역할 및 AWS 암호를 생성하십시오. 자세한 내용은 을 "IAM 역할 및 AWS Secret을 생성합니다"참조하십시오.
2. EKS Kubernetes 클러스터에서 * Add-ons * 탭으로 이동합니다.



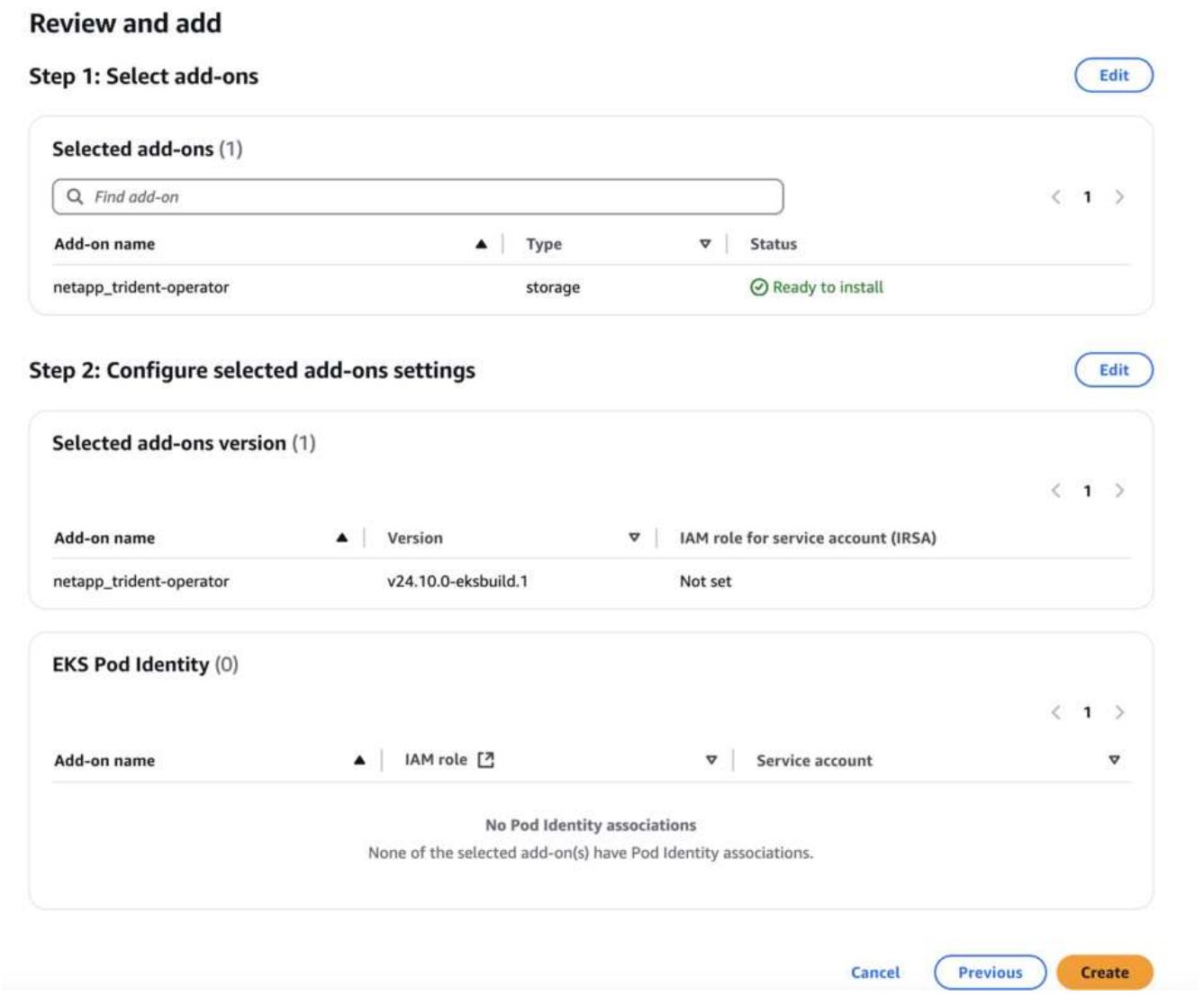
3. AWS Marketplace 애드온 * 으로 이동하여 _STORAGE_CATEGORY를 선택합니다.



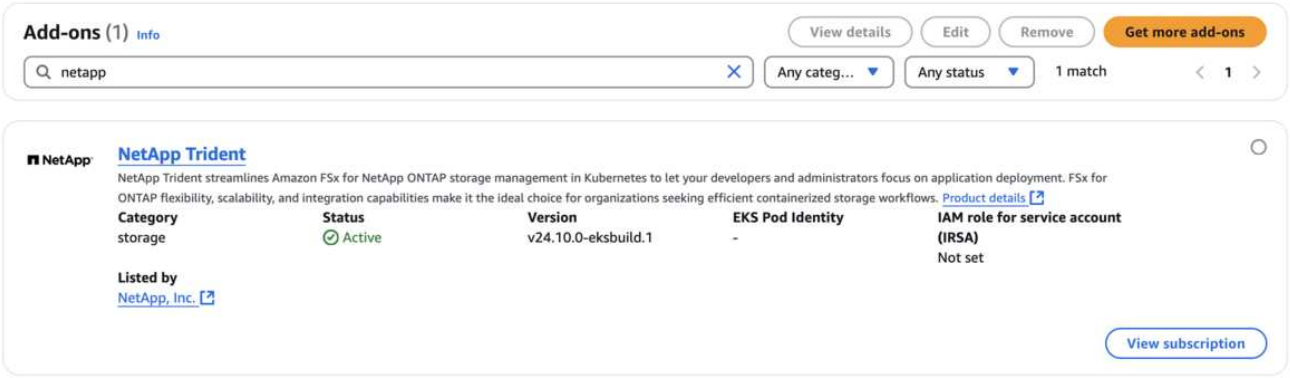
4. NetApp Trident * 를 찾아 Trident 애드온의 확인란을 선택하고 * 다음 * 을 클릭합니다.
5. 원하는 추가 기능 버전을 선택합니다.



6. 필요한 추가 기능 설정을 구성합니다.



7. IRSA(서비스 계정에 대한 IAM 역할)를 사용하는 경우 추가 구성 단계를 참조하세요."여기" .
8. Create * 를 선택합니다.
9. 애드온의 상태가 _Active_인지 확인합니다.



10. 다음 명령을 실행하여 Trident가 클러스터에 올바르게 설치되어 있는지 확인합니다.

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

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

CLI를 사용하여 Trident EKS 애드온을 설치/제거합니다

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

다음 예제 명령은 Trident EKS 추가 기능을 설치합니다.
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (전용 버전 포함)

다음 예시 명령은 Trident EKS 애드온 버전 25.6.1을 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** 애드온을 제거합니다**.**

다음 명령을 실행하면 Trident EKS 추가 기능이 제거됩니다.

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

kubectl로 백엔드를 만듭니다

백엔드는 Trident와 스토리지 시스템 간의 관계를 정의합니다. Trident는 해당 스토리지 시스템과 통신하는 방법과 Trident가 스토리지 시스템에서 볼륨을 프로비저닝하는 방법을 알려줍니다. Trident를 설치한 후 다음 단계는 백엔드를 생성하는 것입니다. `TridentBackendConfig`CRD(사용자 지정 리소스 정의)를 사용하면 Kubernetes 인터페이스를 통해 Trident 백엔드를 직접 생성하고 관리할 수 있습니다. 이 작업은 또는 Kubernetes 배포에 해당하는 CLI 툴을 사용하여 수행할 수 있습니다 `kubectl.

TridentBackendConfig

`TridentBackendConfig`(`tbc` `tbconfig` `tbackendconfig`, ,)는 을 사용하여 Trident 백엔드를 관리할 수 있는 프런트 엔드 이름 기반 CRD `kubectl`입니다. 이제 Kubernetes 및 스토리지 관리자는 전용 명령줄 유틸리티 없이 Kubernetes CLI를 통해 직접 백엔드를 만들고 관리할 수 (`tridentctl`있습니다.

'트리펜엔드구성' 객체가 생성되면 다음과 같은 현상이 발생합니다.

- 백엔드는 사용자가 제공하는 구성에 따라 Trident에 의해 자동으로 생성됩니다. 이 값은 내부적으로 a `TridentBackend` (`tbe`, `tridentbackend`) CR로 표시됩니다.

- 는 `TridentBackendConfig` Trident에서 만든 에 고유하게 바인딩됩니다. `TridentBackend`

각각의 트리젠백엔드구성은 트리젠백엔드(트리젠백엔드)를 통해 일대일 매핑을 유지합니다. 전자는 백엔드를 설계 및 구성하기 위해 사용자에게 제공되는 인터페이스이며, 후자는 Trident가 실제 백엔드 객체를 나타내는 방법입니다.

경고	`TridentBackend` CRS는 Trident에 의해 자동으로 생성됩니다. 수정할 수 없습니다. 백엔드를 업데이트하려면 객체를 수정하여 이 작업을 `TridentBackendConfig` 수행합니다.

'트리엔백엔드구성' 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
```

의 예를 살펴볼 수도 있습니다 "Trident - 장착 공구" 원하는 스토리지 플랫폼/서비스의 샘플 구성을 위한 디렉토리입니다.

를 클릭합니다 `spec` 백엔드 관련 구성 매개 변수를 사용합니다. 이 예에서는 백엔드에서 를 사용합니다 `ontap-san` 여기에 표로 제공된 구성 매개 변수를 사용하여 스토리지 드라이버를 다운로드합니다. 원하는 스토리지 드라이버에 대한 구성 옵션 목록은 을 참조하십시오 "스토리지 드라이버에 대한 백엔드 구성 정보입니다".

이번 시기에는 트리젠백엔드Config CR에 새로 도입된 자격 증명과 delitionPolicy 필드가 포함되어 있습니다.

- "credentials": 이 매개변수는 필수 필드이며 스토리지 시스템/서비스를 인증하는 데 사용되는 자격 증명을 포함합니다. 사용자 생성 Kubernetes Secret으로 설정됩니다. 자격 증명을 일반 텍스트로 전달할 수 없으며 오류가 발생합니다.
- "설명 정책": 이 필드는 트리젠BackendConfig가 삭제될 때 발생하는 동작을 정의합니다. 다음 두 가지 값 중 하나를 사용할 수 있습니다.
  - 삭제(Delete): 이 경우 트리엔백엔드Config CR과 관련 백엔드가 모두 삭제됩니다. 이 값이 기본값입니다.
  - [Tain]: 트리덴트Config CR이 삭제되면 백엔드 정의가 계속 존재하고 tridentctl로 관리될 수 있다. 삭제 정책을 "보존"으로 설정하면 사용자는 이전 릴리스(21.04 이전)로 다운그레이드하고 생성된 백엔드를 유지할 수 있습니다. 이 필드의 값은 '트리엔백엔드구성'이 생성된 후에 업데이트할 수 있습니다.

참고	백엔드 이름은 'pec.backendName'을 사용하여 설정됩니다. 지정하지 않으면 백엔드 이름이 '트리엔백엔드구성' 객체(metadata.name 지정합니다. 'pec.backendName'을 사용하여 백엔드 이름을 명시적으로 설정하는 것이 좋습니다.

팁	로 만든 백엔드에는 `tridentctl` 연결된 `TridentBackendConfig` 개체가 없습니다. CR을 만들어 `TridentBackendConfig` 이러한 백엔드를 관리하도록 선택할 수 `kubectl` 있습니다. 동일한 구성 매개 변수(`spec.storagePrefix`,, `spec.storageDriverName` 등)를 지정할 때 주의를 기울여야 `spec.backendName` 합니다. Trident는 새로 생성된 를 기존 백엔드와 자동으로 바인딩합니다. `TridentBackendConfig`

단계 개요

kubbeck을 사용하여 새 백엔드를 생성하려면 다음을 수행해야 합니다.

1. 를 "쿠버네티스 비밀"생성합니다. 비밀에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. '트리멘백엔드구성' 객체를 만듭니다. 스토리지 클러스터/서비스에 대한 자세한 내용과 이전 단계에서 생성한 암호를 참조하십시오.

백엔드를 생성한 후 "kubbtl get tbc<tbc-name>-n<trident-namespace>"를 사용하여 해당 상태를 관찰하고 추가 세부 정보를 수집할 수 있습니다.

1단계**: Kubernetes Secret** 생성

백엔드에 대한 액세스 자격 증명이 포함된 암호를 생성합니다. 이는 각 스토리지 서비스/플랫폼마다 다릅니다. 예를 들면 다음과 같습니다.

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

이 표에는 각 스토리지 플랫폼의 비밀에 포함되어야 하는 필드가 요약되어 있습니다.

스토리지 플랫폼 암호 필드 설명입니다	비밀	필드 설명입니다
Azure NetApp Files	클라이언트 ID입니다	앱 등록에서 클라이언트 ID
요소(NetApp HCI/SolidFire)	엔드포인트	테넌트 자격 증명이 있는 SolidFire 클러스터의 MVIP입니다
ONTAP	사용자 이름	클러스터/SVM에 연결할 사용자 이름입니다. 자격 증명 기반 인증에 사용됩니다

스토리지 플랫폼 암호 필드 설명입니다	비밀	필드 설명입니다
ONTAP	암호	클러스터/SVM에 연결하는 암호 자격 증명 기반 인증에 사용됩니다
ONTAP	clientPrivateKey를 선택합니다	Base64 - 클라이언트 개인 키의 인코딩된 값입니다. 인증서 기반 인증에 사용됩니다
ONTAP	챕터 사용자 이름	인바운드 사용자 이름입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	챕터시토시크릿	CHAP 이니시에이터 암호입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	chapTargetUsername 을 선택합니다	대상 사용자 이름입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다
ONTAP	챕터타겟이니터시크릿	CHAP 타겟 이니시에이터 암호입니다. useCHAP = TRUE인 경우 필수입니다. ONTAP-SAN과 ONTAP-SAN 경제입니다

이 단계에서 만든 암호는 다음 단계에서 만든 트리젠백엔드Config 개체의 '증명서' 필드에 참조됩니다.

2단계: 을 작성합니다 **`TridentBackendConfig`** 있습니다

이제 '트리엔백구성' CR을 만들 준비가 되었습니다. 이 예에서 'ONTAP-SAN' 드라이버를 사용하는 백엔드는 아래에 나와 있는 ' TridentBackendConfig ' 객체를 사용하여 생성합니다.

```
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**:** 의 상태를 확인합니다 **`TridentBackendConfig`** 있습니다

이제 '트리펜엔드구성' CR을 생성했으므로 상태를 확인할 수 있습니다. 다음 예를 참조하십시오.

```
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 '트리엔백엔드구성' CR에 바인딩되었습니다.

위상은 다음 값 중 하나를 사용할 수 있습니다.

- `Bound`: `TridentBackendConfig` CR은 백엔드에 연결되어 있으며 해당 백엔드에는 가 포함되어 있습니다 `configRef` 로 설정합니다 `TridentBackendConfig` Cr'uid(CR'uid)
- 'Unbound': "로 표현됨. 트리젠백엔드Config 객체가 백엔드에 바인딩되지 않습니다. 새로 만든 트리젠백엔드Config CRS는 기본적으로 이 단계에 있습니다. 단계가 변경된 후에는 다시 바인딩되지 않은 상태로 되돌릴 수 없습니다.
- `Deleting`: `TridentBackendConfig` CR의 `deletionPolicy` 이(가) 삭제되도록 설정되었습니다. 를 누릅니다 `TridentBackendConfig` CR이 삭제되어 삭제 상태로 전환됩니다.
  - 백엔드에 영구 볼륨 클레임(PVC)이 없는 경우 을 `TridentBackendConfig` 삭제하면 Trident이 백엔드와 `TridentBackendConfig` CR을 삭제합니다.
  - 백엔드에 PVC가 하나 이상 있는 경우 삭제 상태로 전환됩니다. 이후 트리젠백엔드Config CR도 삭제 단계로 진입한다. 모든 PVC가 삭제된 후에만 백엔드 및 트리젠백엔드구성이 삭제됩니다.
- 손실: 트리젠백엔드Config CR과 관련된 백엔드가 실수로 또는 고의적으로 삭제되었고, 트리젠백엔드Config CR에는 삭제된 백엔드에 대한 참조가 여전히 있습니다. 이 경우에도 '항목 정책' 값에 관계없이 '트리멘백엔드구성' CR은 삭제할 수 있습니다.
- `Unknown`: Trident가 CR과 연결된 백엔드의 상태 또는 존재를 확인할 수 `TridentBackendConfig` 없습니다. 예를 들어, API 서버가 응답하지 않거나 CRD가 누락된 경우 `tridentbackends.trident.netapp.io` 이 경우 개입이 필요할 수 있습니다.

이 단계에서는 백엔드가 성공적으로 생성됩니다! 다음과 같은 몇 가지 작업을 추가로 처리할 수 있습니다 "백엔드 업데이트 및 백엔드 삭제".

(선택 사항**) 4**단계**:** 자세한 내용을 확인하십시오

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 '트리엔백구성'의 YAML/JSON 덤프를 얻을 수도 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` CR에 대한 응답으로 생성된 백엔드의 `TridentBackendConfig` 및 가 `backendUUID` 포함되어 `backendName` 있습니다. 이 `lastOperationStatus` 필드는 CR의 마지막 작업 상태를 나타냅니다. 이 상태는 `TridentBackendConfig` 사용자가 트리거하거나(예: 사용자가 변경한 내용) Trident에 의해 트리거될 수 있습니다 (예: `spec` Trident 재시작 중). 성공 또는 실패일 수 있습니다. `phase` CR과 백엔드 간의 관계 상태를 `TridentBackendConfig` 나타냅니다. 위의 예에서 에는 `phase` 값이 바인딩되어 있습니다. 즉, CR이 백엔드와 연결되어 있음을 `TridentBackendConfig` 의미합니다.

"kubbctl -n trident tbc <tbc-cr-name>" 명령을 실행하여 이벤트 로그의 세부 정보를 확인할 수 있습니다.

경고	tridentctl을 사용하여 연결된 'TrientBackendConfig' 객체가 포함된 백엔드는 업데이트하거나 삭제할 수 없습니다. tridentctl과 트리멘BackendConfig의 전환 단계를 이해하려면 "여기 를 참조하십시오".

백엔드 관리

kubectl을 사용하여 **Trident** 백엔드를 관리하십시오**.**

kubbtl을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 삭제합니다

를 삭제하면 `TridentBackendConfig` Trident가 백엔드를 삭제/유지하도록 지시합니다(기준 `deletionPolicy`). 백엔드를 삭제하려면 가 `deletionPolicy` DELETE로 설정되어 있는지 확인합니다. 만 삭제하려면 `TridentBackendConfig` 가 유지 로 설정되어 있는지 `deletionPolicy` 확인합니다. 이렇게 하면 백엔드가 계속 존재하고 을 사용하여 관리할 수 `tridentctl` 있습니다.

다음 명령을 실행합니다.

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

Trident은 에서 사용 중인 Kubernetes 비밀을 삭제하지 `TridentBackendConfig` 않습니다. Kubernetes 사용자는 기밀을 정해야 합니다. 비밀 정보를 삭제할 때는 주의해야 합니다. 암호는 백엔드에서 사용하지 않는 경우에만 삭제해야 합니다.

기존 백엔드를 봅니다

다음 명령을 실행합니다.

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

또한 'tridentctl get backend-n trident' 또는 'tridentctl get backend-o YAML-n trident'를 실행하여 존재하는 모든 백엔드 목록을 확인할 수 있습니다. 이 목록에는 tridentctl로 만든 백엔드 또한 포함됩니다.

백엔드를 업데이트합니다

백엔드를 업데이트해야 하는 이유는 여러 가지가 있을 수 있습니다.

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

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

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

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

- 또는 기존 을 변경할 수 있습니다 `TridentBackendConfig` 다음 명령을 사용하는 CR:

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

참고

- 백엔드 업데이트에 실패하면 백엔드는 마지막으로 알려진 구성으로 계속 유지됩니다. 'kubtl get tbc<tbc-name>-o YAML-n trident' 또는 'kubtl t설명해 tbc<tbc-name>-n trident'를 실행하여 로그를 보고 원인을 확인할 수 있습니다.
- 구성 파일의 문제를 확인하고 수정한 후 update 명령을 다시 실행할 수 있습니다.

tridentctl을 사용하여 **Trident** 백엔드를 관리하십시오**.**

tridentctl을 사용하여 백엔드 관리 작업을 수행하는 방법에 대해 알아보십시오.

백엔드를 생성합니다

을 만든 후 "백엔드 구성 파일"에서 다음 명령을 실행합니다.

```
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 확인하고 수정한 후에는 간단히 'create' 명령을 다시 실행할 수 있습니다.

백엔드를 삭제합니다

Trident에서 백엔드를 삭제하려면 다음을 수행합니다.

1. 백엔드 이름 검색:

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

2. 백엔드를 삭제합니다.

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

참고 Trident가 이 백엔드에서 아직 존재하는 볼륨 및 스냅샷을 프로비저닝한 경우 백엔드를 삭제하면 새

> 볼륨이 프로비저닝되지 않습니다. 백엔드는 계속해서 "삭제 중" 상태로 유지됩니다.

기존 백엔드를 봅니다

Trident가 알고 있는 백엔드를 보려면 다음을 실행합니다.

- 요약을 보려면 다음 명령을 실행합니다.

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

- 모든 세부 정보를 보려면 다음 명령을 실행합니다.

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

백엔드를 업데이트합니다

새 백엔드 구성 파일을 생성한 후 다음 명령을 실행합니다.

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

백엔드 업데이트에 실패하면 백엔드 구성에 문제가 있거나 잘못된 업데이트를 시도했습니다. 다음 명령을 실행하여 로그를 보고 원인을 확인할 수 있습니다.

```
tridentctl logs -n trident
```

구성 파일의 문제를 확인하고 수정한 후에는 간단히 'update' 명령을 다시 실행할 수 있습니다.

백엔드를 사용하는 스토리지 클래스를 식별합니다

이것은 JSON으로 백엔드 객체에 대해 tridentctl이 출력하는 질문의 예입니다. 이 유틸리티는 설치해야 하는 JQ 유틸리티를 사용합니다.

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

이는 ' 재젠백엔드구성'을 사용하여 만든 백엔드에도 적용됩니다.

Trident 백엔드 관리 방식을 선택하세요

Trident에서 백엔드를 관리하는 다양한 방법에 대해 알아보십시오.

을 소개합니다 `TridentBackendConfig`이제 관리자는 두 가지 고유한 방식으로 백엔드를 관리할 수 있습니다. 이 질문은 다음과 같습니다.

- tridentctl을 사용하여 만든 백엔드는 트리엔백엔드구성을 사용하여 관리할 수 있습니까?
- 트리덴ctl을 사용하여 트리엔디Config를 사용하여 만든 백엔드를 관리할 수 있습니까?

관리 `tridentctl` 을 사용하여 백엔드를 만듭니다 `TridentBackendConfig`

이 섹션에서는 'Tridentctl' 객체를 만들어 Kubernetes 인터페이스를 통해 직접 생성된 백엔드를 관리하는 데 필요한 단계에 대해 설명합니다.

이 내용은 다음 시나리오에 적용됩니다.

- 가 없는 기존 백엔드가 있습니다 `TridentBackendConfig` 을 사용하여 생성되었기 때문입니다 `tridentctl`.
- tridentctl로 만든 새 백엔드와 다른 트리젠BackendConfig 개체가 있습니다.

두 시나리오 모두에서 Trident의 볼륨 예약 및 운영 기능과 함께 백엔드가 계속 표시됩니다. 관리자는 다음 두 가지 옵션 중 하나를 선택할 수 있습니다.

- tridentctl을 사용하여 만든 백엔드를 관리하려면 계속 사용합니다.
- tridentctl을 사용하여 만든 백엔드를 새 트리젠BackendConfig 개체에 바인딩합니다. 이를 통해 뒷골은 트리덴틀이 아니라 쿠벤틀로 관리된다는 뜻이다.

kubbeck을 사용하여 기존 백엔드를 관리하려면 기존 백엔드에 바인딩되는 '트리젠백엔드구성'을 생성해야 합니다. 작동 방식에 대한 개요는 다음과 같습니다.

1. Kubernetes 암호를 생성하십시오. 비밀에는 Trident가 스토리지 클러스터/서비스와 통신하는 데 필요한 자격 증명이 포함되어 있습니다.
2. '트리멘백엔드구성' 객체를 만듭니다. 스토리지 클러스터/서비스에 대한 자세한 내용과 이전 단계에서 생성한 암호를 참조하십시오. 동일한 구성 매개 변수(예: 'pec.backendName', 'pec.storagePrefix', 'pec.storageDriverName' 등)를 지정할 때는 주의해야 합니다. '현재 백엔드 이름'을 설정해야 합니다.

단계 **0:** 백엔드를 식별합니다

을(를) 생성합니다 `TridentBackendConfig` 이 경우 기존 백엔드에 바인딩되므로 백엔드 구성을 확보해야 합니다. 이 예에서는 다음과 같은 JSON 정의를 사용하여 백엔드를 생성했다고 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 것처럼 백엔드에 대한 자격 증명이 포함된 암호를 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: **A**를 작성합니다 **`TridentBackendConfig`** 있습니다

다음 단계는 기존 ONTAP-NAS-백엔드에 자동으로 바인딩되는 '트리젠백엔드구성' CR을 생성하는 것입니다(예:). 다음 요구 사항이 충족되는지 확인합니다.

- 같은 백엔드 이름은 'sepec.backendName'에 정의되어 있습니다.
- 구성 매개 변수는 원래 백엔드와 동일합니다.
- 가상 풀(있는 경우)은 원래 백엔드와 동일한 순서를 유지해야 합니다.
- 자격 증명은 일반 텍스트가 아닌 Kubernetes Secret을 통해 제공됩니다.

이 경우 트리젠백엔드구성은 다음과 같습니다.

```
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: 의 상태를 확인합니다 **`TridentBackendConfig`** 있습니다

트리젠백엔드구성이 만들어지면 그 단계는 반드시 '바운드'되어야 한다. 또한 기존 백엔드의 백엔드 이름과 UUID도 동일하게 반영되어야 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' 트리펜엔드구성 객체를 사용하여 완벽하게 관리됩니다.

관리 **TridentBackendConfig** 을 사용하여 백엔드를 만듭니다 **tridentctl**

트리덴ctl은 트리엔백구성(TrientBackendConfig)을 사용하여 만든 백엔드를 나열하는 데 사용할 수 있습니다. 또한 관리자는 트리텐틀Config를 삭제하고 pec.deletionPolicy` 가 "Stain"으로 설정되어 있는지 확인하여 tridentctl을 통해 이러한 백엔드를 완벽하게 관리할 수도 있습니다.

단계 **0:** 백엔드를 식별합니다

예를 들어, 다음 백엔드가 ``트리엔백구성”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 해당 결과가 표시됩니다 `TridentBackendConfig` 성공적으로 생성되었으며 백엔드에 바인딩되었습니다 [백엔드의 UUID 관찰].

1단계: 확인 **`deletionPolicy`** 가 로 설정되어 있습니다 **`retain`**

의 가치를 `deletionPolicy` 살펴보겠습니다. 이 설정은 로 `retain`**설정해야 합니다. 이렇게 하면** `CR`**이 삭제되어도** `TridentBackendConfig` 백엔드 정의가 계속 존재하며 로 관리할 수 `tridentctl` 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
```

참고 '정책'이 '유지'로 설정되어 있지 않으면 다음 단계로 진행하지 마십시오.

2단계: 를 삭제합니다 **`TridentBackendConfig`** 있습니다

마지막 단계는 트리엔디Config CR을 삭제하는 것이다. '정책'이 '유지'로 설정되어 있는지 확인한 후 삭제를 계속 수행할 수 있습니다.

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

```
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

오브젝트가 삭제되면 `TridentBackendConfig` Trident은 백엔드 자체를 실제로 삭제하지 않고 기존 오브젝트를 단순히 제거합니다.

스토리지 클래스를 생성하고 관리합니다

Trident 스토리지 클래스를 생성합니다

Kubernetes StorageClass 개체를 구성하고 스토리지 클래스를 생성하여 Trident에 볼륨 프로비저닝 방법을 지시합니다.

Kubernetes StorageClass 개체를 구성합니다

는 "Kubernetes StorageClass 객체" Trident를 해당 클래스에 사용되는 Provisioner로 식별하고 Trident에 볼륨 프로비저닝 방법을 지시합니다. 예를 들면 다음과 같습니다.

```
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
```

Trident에서 볼륨을 프로비저닝하는 방법을 제어하는 데 사용되는 및 매개 변수와 스토리지 클래스가 상호 작용하는 방법에 대한 자세한 `PersistentVolumeClaim` 내용은 을 "Kubernetes 및 Trident 오브젝트"참조하십시오.

스토리지 클래스를 생성합니다

StorageClass 객체를 생성한 후 스토리지 클래스를 생성할 수 있습니다. 보관 클래스 샘플 에는 사용하거나 수정할 수 있는 몇 가지 기본 샘플이 나와 있습니다.

단계

1. Kubernetes 오브젝트이므로 를 사용하십시오 `kubectl` Kubernetes에서 생성해야 합니다.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 이제 Kubernetes와 Trident 모두에서 * BASIC-CSI * 스토리지 클래스가 표시되고 Trident는 백엔드에서 풀을 검색해야 합니다.

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

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

```
./tridentctl -n trident get storageclass basic-csi -o json
```


```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```


보관 클래스 샘플

Trident가 "특정 백엔드에 대한 간단한 스토리지 클래스 정의"제공합니다.

또는 편집할 수 있습니다 `sample-input/storage-class-csi.yaml.templ` 설치 프로그램과 함께 제공되는

파일 및 대치 *BACKEND_TYPE* 스토리지 드라이버 이름을 사용합니다.

```
./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
```

Trident 스토리지 클래스 관리

기존 스토리지 클래스를 보고, 기본 스토리지 클래스를 설정하고, 스토리지 클래스 백엔드를 식별하고, 스토리지 클래스를 삭제할 수 있습니다.

기존 스토리지 클래스를 봅니다

- 기존 Kubernetes 스토리지 클래스를 보려면 다음 명령을 실행합니다.

```
kubectl get storageclass
```

- Kubernetes 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행합니다.

```
kubectl get storageclass <storage-class> -o json
```

- Trident의 동기화된 스토리지 클래스를 보려면 다음 명령을 실행합니다.

```
tridentctl get storageclass
```

- Trident의 동기화된 스토리지 클래스 세부 정보를 보려면 다음 명령을 실행합니다.

```
tridentctl get storageclass <storage-class> -o json
```

기본 스토리지 클래스를 설정합니다

Kubernetes 1.6에는 기본 스토리지 클래스를 설정하는 기능이 추가되었습니다. 사용자가 영구 볼륨 클레임(PVC)에 영구 볼륨을 지정하지 않는 경우 영구 볼륨을 프로비저닝하는 데 사용되는 스토리지 클래스입니다.

- 스토리지 클래스 정의에서 주석 'torageclass.Kubernetes.io/is-default-class'를 true로 설정하여 기본 스토리지 클래스를 정의합니다. 사양에 따라 다른 값이나 주석 부재는 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"}}}'
```

또한 Trident 설치 프로그램 번들에는 이 주석을 포함하는 예제도 있습니다.

참고	클러스터에는 한 번에 하나의 기본 스토리지 클래스만 있어야 합니다. Kubernetes에서 둘 이상의 작업을 수행하는 것을 기술적으로 금지하지는 않지만 기본 스토리지 클래스가 없는 것처럼 동작합니다.

스토리지 클래스에 대한 백엔드를 식별합니다

다음은 Trident 백엔드 객체에 대해 출력하는 JSON으로 답변할 수 있는 질문의 `tridentctl` 예입니다. 이 `jq` 유틸리티는 먼저 설치해야 할 수 있는 유틸리티를 사용합니다.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

스토리지 클래스를 삭제합니다

Kubernetes에서 스토리지 클래스를 삭제하려면 다음 명령을 실행합니다.

```
kubectl delete storageclass <storage-class>
```

'<storage-class>'은(는) 스토리지 클래스로 교체해야 합니다.

이 스토리지 클래스를 통해 생성된 영구 볼륨은 변경되지 않으며 Trident에서 계속 관리합니다.

참고 | Trident는 생성한 볼륨에 대해 빈 값을 `fsType` 적용합니다. iSCSI 백엔드의 경우 StorageClass에 적용하는 것이 좋습니다 `parameters.fsType`. 기존 StorageClasses를 삭제하고 지정된 대로 다시 `parameters.fsType` 생성해야 합니다.

볼륨을 프로비저닝하고 관리합니다

Trident 볼륨 프로비저닝

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

개요

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

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

PVC를 생성한 후 포드에 볼륨을 마운트할 수 있습니다.

PVC를 작성합니다

단계

1. PVC를 작성합니다.

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

2. PVC 상태를 확인합니다.

```
kubectl get pvc
```

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

1. 볼륨을 Pod에 마운트합니다.

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

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

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

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

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

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

샘플 매니페스트

▼ **PersistentVolumeClaim** 샘플 매니페스트

이러한 예는 기본적인 PVC 구성 옵션을 보여줍니다.

RWO 액세스 **PVC**

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

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

NVMe/TCP가 있는 **PVC**

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

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

▼ **POD** 매니페스트 샘플

이 예는 PVC를 포드에 부착하기 위한 기본 구성을 보여줍니다.

기본 구성

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

기본 **NVMe/TCP** 구성

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Trident에서 볼륨을 프로비저닝하는 방법을 제어하는 데 사용되는 및 매개 변수와 스토리지 클래스가 상호 작용하는 방법에 대한 자세한 `PersistentVolumeClaim` 내용은 을 "Kubernetes 및 Trident 오브젝트"참조하십시오.

볼륨 이동

Trident 26.06 버전부터 스토리지 관리자는 워크로드가 계속 실행되는 동안에도 iSCSI 볼륨을 동일한 ONTAP 클러스터 내의 다른 ONTAP 노드 및 애그리게이트로 중단 없이 이동할 수 있습니다. `TridentVolumeMove` 사용자 지정 리소스 또는 `tridentctl` 이와 동등한 기능을 사용하여 이동을 선언적으로 요청하면 Trident가 ONTAP과 호스트를 조정하여 I/O가 중단되지 않도록 합니다.

요구 사항

볼륨 이동 기능은 `ontap-san` iSCSI 볼륨용 드라이버에만 적용됩니다. NVMe 및 NAS 프로토콜은 지원되지 않습니다.

볼륨을 이동하기 전에 다음 요구 사항이 충족되었는지 확인하십시오.

- 백엔드는 ONTAP REST API와 함께 `ontap-san` 드라이버를 사용합니다.
- ONTAP 클러스터는 최소 두 개의 노드로 구성됩니다.
- 귀하는 다음 자격 증명 중 하나를 보유하고 있습니다.
  - 전용 사용자 지정 ONTAP 역할을 사용하는 경우 ONTAP 9.19.1 이상이 필요합니다.
  - 이전 ONTAP 릴리스의 클러스터 관리자 자격 증명입니다.

작동 원리

Trident는 동일한 ONTAP 클러스터 내의 두 위치 간에 볼륨(LUN)을 이동합니다. 데이터 이동 전에 Trident는 기존 경로와 함께 대상에 새로운 iSCSI 경로를 추가하여 호스트 다중 경로 장치가 액세스 권한을 잃지 않도록 합니다. ONTAP이 이동을 완료하면 Trident는 더 이상 사용하지 않는 소스 경로를 제거합니다.

Trident는 이동을 진행하면서 `TridentVolumeMove` 리소스에서 확인할 수 있는 일련의 순서 있는 상태를 거치므로 이동 진행 상황을 추적할 수 있습니다.

볼륨 이동

볼륨 및 위치 식별

1. 이동할 PV의 이름을 찾으세요. PV 이름은 이동 리소스의 이름과 동일합니다.
2. ONTAP 클러스터에서 해당 볼륨의 현재(소스) 노드와 애그리게이트를 확인하고 대상(타겟) 노드와 애그리게이트를 선택합니다.

이동 시작

두 가지 방법 중 하나를 사용하십시오. 두 방법 모두 동일한 `TridentVolumeMove` 리소스를 생성하며, 해당 리소스의 이름은 PV 이름입니다.

tridentctl 사용

```
tridentctl create volume-move \
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \
```

```
  --source-node ontap-node-01 \
  --source-pool aggr1_node01 \
  --target-node ontap-node-02 \
  --target-pool aggr1_node02 \
  -n trident
```

`tvm`는 `volume-move`의 별칭입니다. 이동이 성공하면 리소스를 자동으로 제거하려면 `--delete-after-success 10m`를 추가하세요.

TridentVolumeMove 리소스 사용

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeMove
metadata:
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671   # the PV to move
  namespace: trident
spec:
  sourceNode: ontap-node-01
  sourcePool: aggr1_node01
  targetNode: ontap-node-02
  targetPool: aggr1_node02
  # deleteAfterSuccess: 10m   # optional
```

리소스를 적용합니다:

```
kubectl apply -f tridentvolumemove.yaml
```

사양 필드

sourceNode / targetNode

ONTAP 노드 이름(UUID가 아님).

sourcePool / targetPool

ONTAP 애그리게이트 이름.

deleteAfterSuccess

선택적 지속 시간(예: `30s` 또는 `10m`). `0`로 설정하면 리소스를 즉시 삭제하고, 생략하면 리소스를 유지합니다.

이동 상황을 모니터링합니다

`tvm` 단축 이름 또는 `tridentvolumemove`을(를) 사용하여 진행 상황을 추적합니다.

```
kubectl get tridentvolumemove -n trident
```

현재 상태 및 노드별 진행 상황을 확인하려면 리소스를 설명합니다.

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

볼륨이 `Moving` 상태 동안 동시에 여러 연결이 발생할 수 있습니다. ONTAP에서 대용량 볼륨의 이동을 완료하는 데 시간이 걸릴 수 있기 때문입니다. 원활한 이동을 보장하기 위해 Trident는 Trident 컨트롤러와 Trident 노드 간의 조정을 수행합니다. 각 상태는 다음 단계를 순서대로 수행하며, `Succeeded` 또는 `Failed`으로 종료됩니다.

상태	무슨 일이 일어나고 있습니까?
`Pending`	Trident는 이전 요청을 승인하고 유효성을 검사한 후 스토리지 변경 작업을 시작하기 전에 필요한 정보를 수집합니다.
`ControllerStaging`	Trident가 대상 위치를 준비합니다. 대상 노드의 스토리지 인터페이스를 LUN에 매핑하고 Kubernetes를 업데이트하여 향후 노드 재시작 시 올바른 위치에 다시 연결되도록 합니다.
`NodeStaging`	볼륨이 마운트된 각 호스트에서 Trident는 대상 노드의 새 스토리지 경로를 기존 경로와 함께 추가하여 Pod의 I/O가 두 경로 세트 모두에서 계속되도록 합니다.
`Moving`	ONTAP는 볼륨 데이터를 대상 애그리게이트로 복사하고 전송합니다. 이 단계가 가장 오래 걸리며, ONTAP가 전적으로 처리하는 동안 Trident는 완료될 때까지 폴링합니다. 이 상태 동안에도 새로운 게시가 허용됩니다.
`ControllerUnstaging`	데이터 이동이 완료되었습니다. Trident는 Kubernetes를 업데이트하여 원하는 상태에서 소스 경로를 삭제한 다음 소스 노드의 스토리지 인터페이스를 LUN에서 매핑 해제합니다.
`NodeUnstaging`	각 호스트에서 Trident는 오래된 소스 경로를 제거합니다. 호스트에는 대상 경로만 남게 되고, 다중 경로 장치는 중단 없이 I/O 작업을 계속 수행합니다.
`Succeeded`	볼륨이 대상 위치로 완전히 이동되어 온라인 상태가 되었습니다. Trident는 `TridentVolumeMove` 리소스를 유지하거나, `deleteAfterSuccess`을 설정하면 자동으로 삭제합니다.
`Failed`	복구할 수 없는 오류로 인해 이동 작업이 중단되었거나 시작되지 못했습니다. Trident는 정상적인 작업이 재개될 수 있도록 볼륨 상태를 재설정하고 오류 메시지와 함께 리소스를 유지합니다. 메시지를 검토하고 근본 원인을 해결한 후 리소스를 삭제하고 다시 시도하십시오.

이 `.status.attachments` 목록은 연결된 각 노드의 진행 상황을 해당 `state` 필드에 표시합니다.

첨부 상태	의미
`Pending`	노드가 연결된 것으로 확인되었으며, 작업은 아직 시작되지 않았습니다.
`Bridged`	기존 경로와 함께 새로운 대상 경로가 추가되었습니다.
`Migrated`	이제 새 경로를 통해 트래픽이 제공됩니다.
`Cleaned`	노드에서 오래된 소스 경로가 제거되었습니다.

첨부 상태	의미
`Detached`	노드가 이동 중에 분리되어 Trident가 연결 상태가 다음 단계로 진행될 준비가 되었는지 평가하는 동안 추가 고려 대상에서 제외되었습니다.
`Failed`	이 노드에서 터미널 오류가 발생했습니다. 메시지를 확인하세요.

이동 리소스를 정리합니다

`deleteAfterSuccess`을 설정하면 `Trident`가 리소스를 자동으로 제거합니다. 그렇지 않으면 수동으로 삭제할 수 있습니다. 동일한 볼륨에 대해 다른 이동을 요청하기 전에 해당 볼륨에 대한 `TridentVolumeMove` 리소스가 `Succeeded` 또는 `Failed` 상태로 존재하지 않는지 확인하십시오.

참고 Trident는 실패한 이동을 자동으로 삭제하지 않습니다. Trident는 `Failed`리소스를 감사 기록으로 보관합니다. 근본 원인을 해결한 후에는 다시 시도하기 전에 리소스를 수동으로 삭제하십시오.

Trident 볼륨 확장

Trident Kubernetes 사용자에게 볼륨을 생성한 후에 볼륨을 확장할 수 있는 기능을 제공합니다. iSCSI, NFS, SMB, NVMe/TCP 및 FC 볼륨을 확장하는 데 필요한 구성에 대한 정보를 찾아보세요.

iSCSI 볼륨을 확장합니다

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

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

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

StorageClass 정의를 편집하여 를 설정합니다 `allowVolumeExpansion` 필드를 눌러 로 이동합니다 `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 allowVolumeExpansion 매개변수를 포함하도록 편집합니다.

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

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

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

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

Trident는 영구 볼륨(PV)을 만들고 이 영구 볼륨 클레임(PVC)과 연결합니다.

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

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

3단계: **PVC**를 부착하는 **POD**를 정의합니다

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

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

이 예제에서는 'AN-PVC'를 사용하는 POD가 생성됩니다.

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

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

4단계: **PV**를 확장합니다

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

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
```

```
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

5단계: 확장 확인

PVC, PV 및 Trident 볼륨의 크기를 확인하여 팽창이 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 볼륨을 확장합니다

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

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

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

StorageClass 정의를 편집하여 를 설정합니다 `allowVolumeExpansion` 필드를 눌러 로 이동합니다 `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

이미 존재하는 StorageClass의 경우 allowVolumeExpansion 매개변수를 포함하도록 편집합니다.

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

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

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

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

Trident는 영구 볼륨(PV)을 만들고 이 영구 볼륨 클레임(PVC)과 연결합니다.

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
```

```
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                        CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

3단계**: PVC**를 부착하는 **POD**를 정의합니다

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

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

이 예제에서는 'AN-PVC'를 사용하는 POD가 생성됩니다.

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

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

4단계: PV를 확장합니다

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

```
kubectl edit pvc san-pvc
```

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

5단계: 확장 확인

PVC, PV 및 Trident 볼륨의 크기를 확인하여 팽창이 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 볼륨을 확장합니다

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

1단계: 볼륨 확장을 지원하도록 StorageClass를 구성합니다

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

```
cat storageclass-ontapnas.yaml
```

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

이 옵션 없이 스토리지 클래스를 이미 생성한 경우 'kubeck edit storageclass'를 사용하여 기존 스토리지 클래스를 편집하여 볼륨을 확장할 수 있습니다.

2단계: 생성한 StorageClass를 사용하여 PVC를 생성합니다

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

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

Trident 이 PVC에 대해 20MiB NFS PV를 생성해야 합니다.

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

3단계: **PV**를 확장합니다

새로 생성된 20 MiB PV를 1 GiB로 크기를 조정하려면 PVC를 편집하고 설정하세요.
`spec.resources.requests.storage` 1GiB까지:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
```

```
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

4단계: 확장을 확인합니다

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

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

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
```

```
+----------+--------------------------------------+--------+---------+
|                   NAME                    |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+--------------
+----------+--------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+--------------
+----------+--------------------------------------+--------+---------+
```

RWX NVMe 하위 시스템 제한 사항 이해

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

64개 노드 제한 이해

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

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

이 제한 사항은 NVMe 프로토콜을 사용하는 RWX 볼륨에만 적용됩니다.

NVMe 하위 시스템 모델 이해

볼륨별 하위 시스템 모델(Trident 26.02 이전 릴리스)

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

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

슈퍼 서브시스템 모델(Trident 26.02에서 도입)

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

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

각 RWX NVMe 볼륨은 최대 64개의 노드를 지원합니다.

오류 증상 파악

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

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

이 오류는 ONTAP NVMe 서브시스템 컨트롤러 제한에 도달했음을 나타냅니다.

하위 시스템 제한 오류 해결

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

Trident를 업그레이드하여 슈퍼 서브시스템 모델을 적용합니다

RWX NVMe 볼륨에 슈퍼 서브시스템 모델을 적용하려면 다음을 수행합니다.

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

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

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

컨트롤러 확장성

Trident는 여러 스토리지 드라이버에 걸쳐 동시성을 향상시켜 컨트롤러 확장성을 개선합니다. 컨트롤러 확장성을 활성화하면 Trident 컨트롤러는 스토리지 작업을 직렬화하는 대신 병렬로 처리하여 동시 작업이 많은 Kubernetes 환경에서 처리량을 높입니다.

배포하기 전에, 일반 출시 버전에서 컨트롤러 확장성을 지원하는 Trident 드라이버와 Trident 26.10에서 기술 미리 보기로 제공되는 드라이버를 확인하십시오. 이를 통해 정보에 기반한 배포 결정을 내리고 위험을 관리할 수 있습니다. 컨트롤러 확장성은 기본적으로 비활성화되어 있습니다.

핵심 개념 및 정의

컨트롤러 확장성

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

컨트롤러 확장성을 활성화하면 서로 다른 볼륨 및 백엔드에 대한 작업이 동시에 진행됩니다. 이를 통해 동시 PersistentVolumeClaim 및 VolumeSnapshot 작업이 많은 환경에서 처리량이 증가하고 엔드 투 엔드 작업 시간이 단축됩니다.

기본 동작(직렬 모드)

기본적으로 Trident 컨트롤러는 작업을 한 번에 하나씩 처리합니다. 생성, 삭제, 크기 조정 또는 스냅샷 요청이 모두 완료된 후에야 Trident는 다음 작업을 시작합니다. 직렬 모드는 모든 설치에서 기본적으로 지원되는 모드이며 구성이 필요하지 않습니다.

대부분의 워크로드에는 직렬 모드가 충분합니다. 직렬 처리 시 작업량이 누적되어 처리 지연이 발생하는 경우에만 컨트롤러 확장성을 활성화하십시오.

컨트롤러 확장성 지원

Trident는 다양한 스토리지 드라이버에 대해 서로 다른 성숙도 수준의 컨트롤러 확장성을 지원합니다.

일반 가용성

다음 드라이버는 Trident 26.10의 정식 출시 버전에서 컨트롤러 확장성을 지원합니다.

- 'ONTAP-SAN'
- 'ONTAP-NAS'
- ONTAP-NAS-이코노미
- 'ONTAP-NAS-Flexgroup'
- ONTAP-SAN-이코노미
- `google-cloud-netapp-volumes`
- 'Azure-NetApp-파일'
- '솔더불-산'

참고	The `google-cloud-netapp-volumes`와 `google-cloud-netapp-volumes-san`드라이버는 서로 다릅니다. Trident는 `google-cloud-netapp-volumes`만 지원합니다. 백엔드 구성이나 예제에서 `google-cloud-netapp-volumes-san`를 사용하지 마십시오.

기술 미리 보기

다음 드라이버는 Trident 26.10의 기술 미리 보기 기능으로 컨트롤러 확장성을 지원합니다.

- `asa-r2` (SAN 및 NVMe)

이 드라이버에는 다음과 같은 제한 사항이 있습니다.

- 컨트롤러 동시성 기능은 평가 및 테스트 목적으로만 사용할 수 있습니다.
- 동작 방식은 릴리스마다 변경될 수 있습니다.
- NetApp은 운영 환경에서의 사용을 권장하지 않습니다.

enableConcurrency 작동 방식

이 설정 `enableConcurrency`을 `true`로 설정하면 Trident는 컨트롤러가 관리하는 모든 백엔드에 걸쳐 동시 처리를 적용합니다. 이 설정은 모든 백엔드에 동시에 적용됩니다. 개별 백엔드 또는 개별 드라이버에 대해서는 활성화할 수 없습니다.

구성된 모든 백엔드는 일반 출시 또는 기술 미리 보기 목록에 있는 드라이버를 사용해야 합니다. 백엔드 중 하나라도 지원되지 않는 드라이버를 사용하는 경우 Trident가 시작되지 않으며, Trident는 지원되지 않는 드라이버를 사용하는 백엔드를 추가하지 않습니다.

시작 기능을 복원하려면 지원되지 않는 드라이버를 사용하는 백엔드를 제거하거나 재구성하거나 컨트롤러 확장성을 비활성화하십시오.

활성화하기 전에

중요함	컨트롤러 확장성을 활성화하기 전에 구성된 모든 백엔드가 일반 공급 또는 기술 미리 보기 목록에 있는 드라이버를 사용하는지 확인하십시오. 백엔드 중 하나라도 지원되지 않는 드라이버를 사용하는 경우, 해당 기능을 활성화한 후 Trident가 시작되지 않습니다.

컨트롤러 확장성 활성화 여부를 결정하려면 다음 표를 참조하십시오.

환경이	그 다음에
지원되는 드라이버만 사용하며 컨트롤러 작업의 백로그가 발생합니다.	컨트롤러 확장성을 활성화합니다.
지원되지 않는 드라이버를 사용합니다	기본 직렬 모드를 유지하십시오. 컨트롤러 확장성을 활성화하지 마십시오.
처리량이 적고 백로그가 없는 상태를 처리합니다.	기본 직렬 모드를 유지하십시오.

컨트롤러 확장성 활성화

이 `enableConcurrency` 구성 옵션은 컨트롤러의 확장성을 제어합니다. Trident 설치 시 또는 기존 배포를 업데이트할 때 이 옵션을 명시적으로 활성화해야 합니다.

Trident 운영자 배포

Trident 오퍼레이터를 사용하여 컨트롤러 확장성을 활성화하려면 `TridentOrchestrator` 사용자 정의 리소스(CR)에서 `enableConcurrency`을(를) `true`(으)로 설정하십시오.

신규 설치

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

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

CR 적용:

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

기존 설치

기존 `TridentOrchestrator` CR에 패치를 적용하여 컨트롤러 확장성을 활성화합니다.

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

Trident가 설정을 적용했는지 확인합니다.

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

Helm 배포

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

신규 설치

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

기존 설치

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

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

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

그런 다음 값 파일을 사용하여 설치 또는 업그레이드하십시오.

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

tridentctl 배포

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

신규 설치

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

기존 설치

기존 `tridentctl` 배포 환경에서 컨트롤러 확장성을 활성화하려면 Trident를 제거한 후 다음 플래그를 사용하여 다시 설치하십시오.

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

컨트롤러 확장성이 활성화되어 있는지 확인하십시오.

컨트롤러 확장성을 활성화한 후 Trident 컨트롤러가 동시 실행 기능을 사용하여 실행되는지 확인하십시오. 컨트롤러 Pod 로그를 확인하십시오.

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

출력에는 동시 실행이 활성화되었음을 확인하는 로그 항목이 포함되어 있습니다.

동시성 동작

컨트롤러 확장성이 활성화되면 Trident 컨트롤러는 다음과 같은 동작을 적용합니다.

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

주의사항 및 제한사항

Trident 26.10의 컨트롤러 확장성과 관련하여 다음과 같은 사항을 고려해야 합니다.

- 컨트롤러 확장성은 일반 출시 및 기술 미리 보기 목록에 있는 드라이버만 지원합니다. 자세한 내용은 활성화하기 전에을 참조하십시오.

- Trident 컨트롤러는 동시 실행을 내부적으로 관리합니다. 이번 릴리스에서는 사용자가 구성할 수 있는 동시 실행 제한을 제공하지 않습니다.
- 전반적인 처리량은 사용 중인 스토리지 드라이버, 백엔드 응답성 및 Kubernetes API 서버 성능에 따라 달라집니다.
- 높은 동시성은 백엔드 스토리지 시스템의 부하를 증가시킬 수 있습니다.
- 컨트롤러 확장성 동작은 모든 드라이버에서 동일하지 않습니다.
- 기술 미리보기 드라이버는 고부하 환경에서 성능이 불안정할 수 있으며, 릴리스 간에 동작이 변경될 수 있습니다.
- 병렬 실행으로 인해 동시 작업 디버깅이 더 복잡해질 수 있습니다. 메트릭과 로그를 통해 작업 출력이 뒤섞여 나타나는 것을 확인할 수 있습니다.

권장 사항

컨트롤러 확장성을 활성화하기 전에 다음 단계를 완료하십시오.

1. 구성된 모든 백엔드가 일반 출시 또는 기술 미리 보기 목록에 있는 드라이버를 사용하는지 확인하십시오.
2. 변경 사항을 프로덕션 환경에 적용하기 전에 비프로덕션 클러스터에서 테스트하십시오.
3. 변경 사항을 적용한 후 컨트롤러 확장성이 활성화되었는지 확인하십시오.

컨트롤러 확장성을 활용할 때는 다음과 같은 일반적인 권장 사항을 적용하십시오.

- 높은 확장성이 요구되는 프로덕션 환경에는 일반 가용성 드라이버를 사용하십시오.
- 기술 미리보기 드라이버는 비운영 환경에서만 평가하십시오.
- 대규모 운영 환경에서 백엔드 및 컨트롤러 성능을 모니터링하십시오.
- 자동화 스크립트에서 작업 순서를 가정하지 마십시오.

자동 볼륨 확장

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

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

Trident Autogrow 정책은 PVC를 확장합니다. 이는 "ONTAP 볼륨 자동 크기 조정"와(과) 다르며, PVC 크기를 변경하지 않고 백킹 FlexVol을 확장합니다.

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

요구 사항

자동 볼륨 확장을 구성하기 전에 다음 요구 사항을 충족하는지 확인하십시오.

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

제한 사항

- ONTAP 9.16.1 이전 버전의 ONTAP Non-Volatile Memory Express 원시 블록 볼륨은 자동 확장을 지원하지 않습니다.
- 스토리지 영역 네트워크 볼륨의 경우, `growthAmount`이(가) 50메비바이트 이하이면 Trident는 크기 조정 전에 자동으로 값을 51메비바이트로 늘립니다. 단, 결과 크기가 `maxSize`을(를) 초과하지 않는 경우에 한합니다.
- 브라운필드 환경에서는 볼륨 게시 마이그레이션 동작으로 인해 특정 기존 볼륨에 대해 자동 확장이 작동하지 않을 수 있습니다.
- 볼륨이 `maxSize`에 도달하면 더 이상 확장되지 않습니다.
- 자동 볼륨 확장을 위해 지원되는 프로토콜:
  - 네트워크 파일 시스템(NFS)
  - Internet Small Computer Systems Interface(iSCSI)
  - 비휘발성 메모리 익스프레스(NVMe)
  - 파이버 채널 프로토콜(FCP)

자동 확장 정책이 적용된 볼륨 프로비저닝

자동 확장 정책은 두 가지 수준에서 구성할 수 있습니다.

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

자동 증가 정책 생성

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

다음 사항을 확인하십시오.

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

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

워크플로의 어느 단계에서든 자동 확장 정책을 생성할 수 있습니다.

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

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

자동 확장 정책 사양

자동 증가 정책은 다음과 같이 정의된 Kubernetes 사용자 지정 리소스입니다.

필드에 입력합니다	설명	형식	필수 요소입니다	예	기본값
이름	고유 정책 식별자	문자열	예	production-db-policy	None
usedThreshold	확장을 트리거하는 용량 비율	백분율 문자열	예	"80%"	None
growthAmount	임계값에 도달했을 때 증가할 양	백분율 또는 크기	아니요	"10%" 또는 "5Gi"	"10%"
maxSize	최대 볼륨 크기 제한	Kubernetes 수량	아니요	"500Gi"	무제한

자동 증가 정책 생성

단계

1. Autogrow Policy를 정의하는 YAML 파일을 생성합니다.

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

2. 클러스터에 정책을 적용합니다.

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

3. 정책이 생성되었는지 확인하십시오.

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

예상 출력

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

정책 상태

정책을 생성하면 Trident가 사양을 검증하고 다음 상태 중 하나를 할당합니다.

상태	설명	조치 필요
성공	정책이 검증되어 사용할 준비가 되었습니다.	없음.
실패했습니다	유효성 검사 오류가 감지되었습니다.	사양을 검토하고 수정하십시오.
삭제 중	삭제가 진행 중입니다.	완료될 때까지 기다립니다.

정책을 **StorageClass**에 연결합니다

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

참고 StorageClass는 다음을 가져야 합니다 `allowVolumeExpansion: true`.

단계

1. Autogrow Policy 어노테이션을 사용하여 StorageClass를 생성하거나 수정합니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. StorageClass를 적용합니다.

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

3. 주석을 확인하세요:

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

예상 출력

```
production-db-policy
```

정책 우선 순위

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

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

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

구성 예

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

운영 데이터베이스에 대한 보수적인 정책

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
```

```
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

고정 증가 단위로 로그 스토리지

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

기본값이 포함된 최소 정책

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

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

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

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

무제한 **maxSize**로 공격적인 성장

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
```

```
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

소수 백분율이 포함된 정책

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

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

자동 확장 기능이 있는 **NAS StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

자동 확장 기능이 있는 **SAN StorageClass**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
```

```
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

ONTAP 볼륨 자동 크기 조정을 구성합니다

영구 볼륨을 지원하는 FlexVol 볼륨에서 ONTAP 볼륨 자동 크기 조정을 활성화하도록 Trident를 구성합니다. 사용된 공간이 구성된 임계값에 도달하면 ONTAP은 FlexVol을 최대 크기까지 자동으로 확장합니다. 이를 통해 FlexVol의 크기를 수동으로 조정하지 않고도 스냅샷 및 스토리지 오버헤드를 위한 추가 용량을 제공합니다.

ONTAP 볼륨 자동 크기 조정은 백킹 FlexVol만 확장하며, PersistentVolumeClaim(PVC) 크기는 변경하지 않습니다. 이는 PVC를 확장하는 "Trident 자동 성장 정책"와(과) 다릅니다.

Trident는 볼륨을 생성, 복제 또는 크기 조정할 때, 그리고 관리하는 볼륨을 가져올 때 백엔드 자동 크기 조정 설정을 적용합니다.

요구 사항

- Trident 26.10 이상 버전.
- 다음 드라이버 중 하나:
  - `ontap-san` iSCSI, NVMe 또는 파이버 채널 프로토콜(FCP)을 지원합니다.
  - `ontap-san-economy` iSCSI를 사용합니다.
- 온프레미스 ONTAP:
  - ONTAP REST API를 사용하는 백엔드의 경우 ONTAP 9.12.1 이상이어야 합니다. NVMe 백엔드는 반드시 ONTAP REST API를 사용해야 합니다.
  - ONTAPI(ZAPI)를 사용하는 백엔드의 경우, Trident에서 지원하는 모든 ONTAP 버전. "Trident 요구사항"을(를) 참조하십시오.

제한 사항

- ONTAP 볼륨 자동 크기 조정은 FlexVols를 축소하지 않습니다.
- 자동 크기 조정은 백엔드별로 구성됩니다. 서로 다른 워크로드에 대해 서로 다른 설정을 사용하려면 별도의 백엔드를 생성하십시오.
- ONTAP 볼륨 자동 크기 조정은 ONTAP NAS 드라이버, ASA r2 시스템, NetApp AFX 또는 기타 분산형 ONTAP 시스템, 또는 Google Cloud NetApp Volumes에서 지원되지 않습니다. 이러한 구성에서는 백엔드 생성이 실패합니다.
- TridentMirrorRelationship 복제 및 MetroCluster 구성은 ONTAP 볼륨 자동 크기 조정과 함께 지원되지 않습니다. Trident는 이러한 구성을 차단하지 않습니다.
- `ontap-san-economy`의 경우, `maximumSize`는 백분율이어야 합니다.
- `ontap-san`의 경우, 절대 `maximumSize`는 크기 제한으로 작용합니다. 기본 FlexVol이 해당 제한을 초과해야 하는 경우 PVC 생성 또는 확장이 실패합니다.

- `ontap-san-economy` FlexVol은 여러 PVC에 대한 LUN을 포함할 수 있습니다. 각 볼륨 작업은 현재 백엔드의 자동 크기 조정 설정을 전체 FlexVol에 적용합니다. 여러 백엔드가 FlexVols를 공유하는 경우, 가장 최근 작업이 해당 설정을 결정합니다. `ontap-san-economy`의 경우, 모든 백엔드가 동일한 기본값을 사용하도록 `mode: grow`만 구성하고 `growThresholdPercent` 및 `maximumSize`는 생략하십시오.

ONTAP 볼륨 자동 크기 조정 옵션

백엔드 구성의 최상위 `ontapAutosize` 블록에 이러한 옵션을 설정합니다.

옵션을 선택합니다	설명	형식	필수 요소입니다	**`ontapAutosize`**이(가) 존재할 때 기본값	예
`mode`	자동 크기 조정을 활성화 또는 비활성화합니다. `grow`는 자동 크기 조정을 활성화하고, `off`는 비활성화합니다.	문자열	아니요	`off`	`grow` 또는 `off`
`growThresholdPercent`	ONTAP이 FlexVol을 확장하는 사용 공간 백분율입니다. 유효 값은 2부터 99까지입니다.	정수	아니요	`85`	`90`
`maximumSize`	최대 FlexVol 크기이며, 해당 크기의 백분율 또는 절대 크기로 표시됩니다. 절대 크기는 `ontap-san`에 대해서만 지원됩니다.	백분율 또는 Kubernetes 수량	아니요	`"20%"`	`"20%"` 또는 `"2Ti"`

`ontapAutosize` 블록이 있는 경우 기본 설정이 적용됩니다. 블록을 생략하면 `Trident`는 `FlexVols`의 자동 크기 조정 설정을 변경하지 않습니다.

Trident는 백분율 `maximumSize`을 절대 크기로 변환하고 볼륨 크기가 조정될 때마다 다시 계산합니다. 예를 들어, `maximumSize: "20%"`를 사용하면 100GiB FlexVol이 120GiB까지 커질 수 있습니다.

ONTAP 볼륨 자동 크기 조정을 구성합니다

1. 백엔드 구성에 `ontapAutosize` 블록을 추가합니다:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: ontap-san-autosize
  managementLIF: 10.0.0.1
  svm: svm_san
  sanType: iscsi
  ontapAutosize:
```

```
    mode: grow
    growThresholdPercent: 85
    maximumSize: "20%"
  credentials:
    name: backend-tbc-ontap-san-secret
```

2. 백엔드를 생성합니다:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

3. 이 백엔드에서 볼륨을 프로비저닝한 후, 해당 Trident 볼륨에 적용된 자동 크기 조정 설정을 확인하십시오:

```
kubectl get tridentvolume <pv-name> -n trident \
  -o jsonpath='{.config.ontapAutosizeMode}
{.config.ontapAutosizeGrowThresholdPercent}
{.config.ontapAutosizeMaximumSize}'
```

위 구성에 대한 예상 출력은 다음과 같습니다.

```
grow 85 20%
```

백엔드에서 자동 크기 조정 설정 업데이트

백엔드를 업데이트하여 `ontapAutosize` 블록을 추가, 변경 또는 제거할 수 있습니다. 이 변경 사항은 기존 FlexVols를 즉시 업데이트하지 않습니다:

- `ontap-san`의 경우 새 볼륨, 클론 및 관리형 가져오기는 새 설정을 사용합니다. 기존 볼륨은 크기 조정 시를 포함하여 현재 설정을 유지합니다.
- `ontap-san-economy`의 경우, 다음에 Trident가 해당 볼륨에서 볼륨을 생성, 크기 조정, 복제 또는 가져올 때 공유 FlexVol은 새 설정을 수신합니다.

구성 예

`ontap-san` FCP 및 절대 최대 크기 사용

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "sanType": "fcp",
```

```
  "ontapAutosize": {
    "mode": "grow",
    "growThresholdPercent": 90,
    "maximumSize": "2Ti"
  }
}
```

ONTAP-SAN-이코노미

이 예제에서는 `mode`만 설정하므로 `Trident`는 `growThresholdPercent` 및 `maximumSize`에 대한 기본값을 사용합니다:

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "ontapAutosize": {
    "mode": "grow"
  }
}
```

자주 묻는 질문

ONTAP 볼륨 자동 크기 조정 기능을 **Trident** 자동 증가 정책과 함께 사용할 수 있습니까**?**

예. Trident 자동 확장 정책은 PVC를 확장하며, Trident는 각 크기 조정 시마다 ONTAP 자동 크기 조정 최대값을 다시 계산합니다. 절대 `maximumSize`을 `ontap-san`와 함께 사용하는 경우, 백업 FlexVol이 해당 제한 내에 유지되도록 정책의 `maxSize`을 충분히 낮게 설정하십시오. Trident가 스냅샷 예약 및 메타데이터를 위한 공간을 추가하기 때문에 FlexVol은 PVC보다 큽니다.

생략 **`ontapAutosize`**과 설정 **`mode: off`**의 차이점은 무엇입니까**?**

`ontapAutosize`을(를) 생략하면 `Trident`는 `ONTAP FlexVols`의 자동 크기 조정 설정을 변경하지 않습니다. `mode: off`을(를) 설정하면 `Trident`는 소스에서 자동 크기 조정 설정을 상속한 클론을 포함하여 생성하거나 업데이트하는 `FlexVols`에서 자동 크기 조정을 비활성화합니다.

PVC 확장이 **`ontapAutosize maximum … is below required FlexVol size`** 오류로 실패한 이유는 무엇입니까**?**

볼륨은 절대값 `maximumSize`을 사용하며, 확장된 `PVC`에는 더 큰 `FlexVol`이 필요합니다. `ontap-san` 볼륨은 생성 시 설정된 최대 크기를 유지하므로 백엔드를 업데이트해도 기존 볼륨의 설정은 변경되지 않습니다. 새 볼륨의 경우 `maximumSize`을 늘리거나 백분율을 사용하십시오.

자동 크기 조정 오류로 인해 볼륨 생성이 실패한 이유는 무엇입니까?

ONTAP이 자동 크기 조정 설정을 거부했으며, PVC 이벤트에 ONTAP의 거부 이유가 포함되어 있습니다. 예를 들어, 최대 크기가 ONTAP 시스템에서 지원하는 최대 FlexVol 크기를 초과할 수 있습니다. 백엔드에서 `maximumSize`를 낮추십시오. Trident는 다음에 볼륨 생성을 시도할 때 업데이트된 값을 사용합니다.

관련 정보

- "Trident Autogrow 정책을 사용한 자동 볼륨 확장"
- "ONTAP 명령어 참조: volume autosize"

자동 증가 정책 관리

자동 증가 정책을 생성한 후에는 해당 정책을 보고, 업데이트하고, 삭제할 수 있습니다. 또한 특정 정책을 사용하는 볼륨을 모니터링할 수도 있습니다.

자동 증가 정책 보기

모든 정책 나열

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

```
kubectl get tridentautogrowpolicy
```

또는 다음을 사용하십시오 `tridentctl`:

```
tridentctl get autogrowpolicy
```

정책 세부 정보 보기

정책의 전체 사양 및 상태를 보려면 다음을 수행합니다.

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

YAML 형식으로 정책과 해당 정책에 연결된 볼륨을 보려면 다음을 수행합니다.

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

자동 증가 정책 업데이트

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

중요함	변경 사항은 해당 정책을 사용하는 모든 볼륨에 영향을 미칩니다. 가능한 경우, 먼저 비운영 환경에서 변경 사항을 테스트하십시오.

단계

1. 정책을 편집합니다.

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

2. `spec` 필드를 필요에 따라 수정합니다.

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

3. 저장하고 종료하세요. 변경 사항은 즉시 적용됩니다.

업데이트 고려 사항

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

자동 증가 정책 삭제

자동 확장 정책은 볼륨이 활발하게 사용되는 동안 실수로 삭제되는 것을 방지하기 위해 파이널라이저 보호 기능을 사용합니다.

단계

1. 정책을 삭제합니다.

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

2. 볼륨이 여전히 해당 정책을 사용하는 경우 삭제 작업이 `Deleting` 상태가 됩니다. 어떤 볼륨이 영향을 받는지 확인하십시오.

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

3. 영향을 받는 각 볼륨에서 정책을 제거합니다. 다음 옵션 중 하나를 선택하십시오.

   ◦ 옵션 **A:** 주석을 `**"none"**`로 설정하여 자동 크기 증가를 명시적으로 비활성화:

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

- 옵션 **B:** 주석을 완전히 제거합니다.

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

삭제 동작

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

자동 증가 정책 사용량 모니터링

정책을 사용하여 볼륨 확인

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

볼륨이 사용하는 정책 찾기

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

정책 이벤트 모니터링

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

지원되는 프로토콜

Autogrow는 다음과 같은 스토리지 프로토콜을 지원합니다:

- NFS 를 참조하십시오
- iSCSI
- FCP

- NVMe

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

지원되는 백엔드

Autogrow는 ONTAP 백엔드를 지원합니다. 또한 Autogrow는 NAS 및 SAN 볼륨 모두에 대해 Google Cloud NetApp Volumes(GCNV)를 지원합니다.

알려진 제한 사항

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

자주 묻는 질문

Trident는 언제 임계값을 평가합니까**?**

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

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

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

볼륨이 이미 프로비저닝된 후에도 정책을 적용할 수 있습니까**?**

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

기존 PVC에 정책을 적용하려면:

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

기존 StorageClass에 정책을 적용하려면 다음을 수행합니다.

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

StorageClass와 PVC 모두에 자동 증가 정책을 설정하면 어떻게 되나요?

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

예를 들어, 다음과 같은 StorageClass가 있습니다.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

그리고 StorageClass 정책을 재정의하는 이 PVC는 다음과 같습니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

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

특정 볼륨의 자동 확장을 비활성화하려면 어떻게 해야 하나요?

PVC 어노테이션을 `"none"`로 설정하십시오. 이 설정은 해당 볼륨에 대한 StorageClass 수준 정책을 재정의합니다.

```
kubectl annotate pvc <pvc-name> \
```

```
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

자동 크기 증가 기능이 비활성화되어 있는지 확인할 수 있습니다.

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

예상 출력

```
none
```

볼륨이 **maxSize**에 도달하면 어떻게 됩니까**?**

Trident 볼륨 확장을 중지합니다. 사용량이 계속 증가하더라도 해당 볼륨에 대한 추가 크기 조정 요청을 생성하지 않습니다 `usedThreshold`.

예를 들어, 이 정책을 사용하면 Trident는 볼륨이 100GiB에 도달한 후 더 이상 볼륨을 늘리지 않습니다.

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

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

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

볼륨을 재시작하지 않고 정책을 변경할 수 있습니까**?**

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

정책을 업데이트하려면 다음을 수행합니다.

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

필요에 따라 필드를 수정합니다.

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

저장 후 종료합니다. 업데이트된 정책을 확인합니다.

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

예상 출력

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

내 정책이 실패 상태인 이유는 무엇입니까?

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

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

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

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

정책을 삭제할 수 없는 이유는 무엇인가요?

정책은 파이널라이저 보호 기능을 사용합니다. 볼륨이 여전히 정책을 사용하는 경우, 삭제 작업은 `Deleting` 상태에 들어가 정책에서 모든 볼륨을 제거할 때까지 기다립니다.

영향을 받는 볼륨을 식별하십시오.

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

그런 다음 각 PVC에서 주석을 제거합니다.

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

모든 볼륨을 제거하면 Trident가 파이널라이저를 해제하고 정책을 삭제합니다.

autogrow는 어떤 백엔드를 지원합니까?

Autogrow는 ONTAP 백엔드를 지원합니다. 또한 NAS 및 SAN 볼륨 모두에 대해 Google Cloud NetApp Volumes(GCNV)도 지원합니다.

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

기존 배포 환경의 기존 볼륨은 자동 확장이 적용되기 전에 볼륨 게시 마이그레이션을 완료해야 할 수 있습니다. 마이그레이션 상태를 확인하려면 Trident 컨트롤러 로그를 확인하십시오.

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

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

NAS 백엔드

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 백엔드

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN 볼륨의 최소 증가량은 얼마입니까?

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

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

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

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

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

Trident 볼륨 가져오기

기존 스토리지 볼륨을 Kubernetes PV로 가져오려면 `tridentctl import`를 사용하거나 Trident 가져오기 어노테이션을 사용하여 영구 볼륨 클레임(PVC)을 생성할 수 있습니다.

개요 및 고려 사항

Trident로 볼륨을 가져올 수 있는 대상:

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

고려 사항

볼륨을 가져오기 전에 다음 고려 사항을 검토하십시오.

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

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

- PVC에 지정해야 하지만 `StorageClass` Trident는 가져오는 동안 이 매개변수를 사용하지 않습니다. 스토리지 클래스는 볼륨 생성 중에 스토리지 특성에 따라 사용 가능한 풀에서 선택하는 데 사용됩니다. 볼륨이 이미 있으므로 가져오는 동안 풀을 선택할 필요가 없습니다. 따라서 볼륨이 PVC에 지정된 스토리지 클래스와 일치하지 않는 백엔드 또는 풀에 있더라도 가져오기에 실패합니다.
- 기존 체적 크기는 PVC에서 결정되고 설정됩니다. 스토리지 드라이버에서 볼륨을 가져온 후 PV는 PVC에 대한 ClaimRef를 사용하여 생성됩니다.
  - 처음에 부가세 반환 청구액 정책이 로 설정되어 있습니다 `retain` PV에서 Kubernetes에서 PVC 및 PV를 성공적으로 바인딩하면 스토리지 클래스의 부가세 반환 청구액 정책에 맞게 부가세 반환 청구액 정책이 업데이트됩니다.
  - 스토리지 클래스의 회수 정책이 `delete`인 경우, Trident는 PV가 삭제될 때 스토리지 볼륨도 삭제합니다.
- 기본적으로 Trident는 PVC를 관리하고 백엔드에서 FlexVol 볼륨과 LUN 또는 qtree의 이름을 변경합니다. 관리되지 않는 볼륨을 가져오려면 `--no-manage` 플래그를 전달하고, 볼륨 이름을 유지하려면 `--no-rename` 플래그를 전달할 수 있습니다.
  - `--no-manage`* - 사용하는 경우 `--no-manage` 플래그가 지정되면 Trident 객체의 수명 주기 동안 PVC 또는 PV에 대한 추가 작업을 수행하지 않습니다. PV가 삭제되어도 저장 볼륨은 삭제되지 않으며 볼륨 복제 및 볼륨 크기 조정과 같은 다른 작업도 무시됩니다.
  - `--no-rename`* - `--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), `ontap-san-economy` 및 `ontap-nas-economy`(qtree 가져오기) 드라이버에서만 지원됩니다.

팁	이러한 옵션은 컨테이너화된 워크로드에 Kubernetes를 사용하지만 그렇지 않은 경우

Kubernetes 외부에서 스토리지 볼륨의 수명 주기를 관리하려는 경우에 유용합니다.

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

볼륨을 가져옵니다

`tridentctl import`를 사용하거나 Trident 가져오기 주석이 포함된 PVC를 생성하여 볼륨을 가져올 수 있습니다.

참고 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
   ```

   참고 필수 매개변수만 포함하십시오. PV 이름이나 볼륨 크기와 같은 추가 매개변수는 가져오기 명령이 실패할 수 있습니다.

2. 사용하다 `tridentctl import` 볼륨을 포함하는 Trident 백엔드의 이름과 스토리지에서 볼륨을 고유하게 식별하는 이름(예: ONTAP FlexVol, Element Volume)을 지정하는 명령입니다. 그만큼 `-f` PVC 파일의 경로를 지정하려면 인수가 필요합니다.

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

PVC 주석 사용

단계

1. 필요한 Trident 가져오기 주석이 포함된 PVC YAML 파일(예: `pvc.yaml`)을 생성합니다. PVC 파일에는 다음 내용이 포함되어야 합니다.

   - `name` 및 `namespace` 메타데이터
   - `accessModes`, `resources.requests.storage` 및 `storageClassName` 사양
   - 주석:
     - `trident.netapp.io/importOriginalName`: 백엔드의 볼륨 이름
     - `trident.netapp.io/importBackendUUID`: 볼륨이 존재하는 백엔드 UUID
     - `trident.netapp.io/notManaged` (선택 사항): 관리되지 않는 볼륨의 경우 `"true"`로 설정합니다. 기본값은 `"false"`입니다.

     다음은 관리형 볼륨을 가져오기 위한 예시 사양입니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. PVC YAML 파일을 Kubernetes 클러스터에 적용합니다.

```
kubectl apply -f <pvc-file>.yaml
```

   Trident는 볼륨을 자동으로 가져와 PVC에 바인딩합니다.

예

지원되는 드라이버에 대한 다음 볼륨 가져오기 예를 검토하십시오.

ONTAP NAS 및 ONTAP NAS FlexGroup를 지원합니다

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

참고

- `ontap-nas-economy` 드라이버를 사용하여 qtree를 가져오려면 NFS qtree 가져오기를 참조하십시오. Trident가 qtree를 가져올 때, qtree를 포함하는 버킷으로 상위 FlexVol도 함께 가져옵니다. qtree는 개별적으로 가져와야 합니다.
- 를 클릭합니다 `ontap-nas` 및 `ontap-nas-flexgroup` 드라이버는 중복 볼륨 이름을 허용하지 않습니다.

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

tridentctl을 사용한 **ONTAP NAS** 예제

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

관리 볼륨

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

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |              BACKEND UUID              | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

관리되지 않는 볼륨

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

다음 예에서는 를 가져옵니다 `unmanaged_volume` 를 누릅니다 `ontap_nas` 백엔드:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |              BACKEND UUID              | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
```

```
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

PVC 주석을 사용한 **ONTAP NAS** 예제

다음 예제는 PVC 주석을 사용하여 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `81abcb27-ea63-49bb-b606-0a5315ac5f21`에서 `ontap_volume1`라는 이름의 `1GiB` `ontap-nas` 볼륨을 가져옵니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

관리되지 않는 볼륨

다음 예제는 PVC 주석을 사용하여 RWO 액세스 모드가 설정된 백엔드 `34abcb27-ea63-49bb-b606-0a5315ac5f34`에서 이름이 `ontap-volume2`인 `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>
```

qtree 가져오기

참고	Qtree 가져오기는 Trident 26.10의 기술 미리보기 기능입니다. 기술 미리보기 기능은 실제 운영 환경에서 사용할 수 없습니다.

`ontap-nas-economy` 드라이버를 사용하여 기존 ONTAP qtree를 Trident로 가져올 수 있습니다. 해당 qtree를 포함하는 백엔드의 UUID와 함께 `flexvol/qtree` 형식의 FlexVol 볼륨 및 qtree 이름으로 qtree를 식별합니다. <<개요 및 고려 사항>>의 일반적인 가져오기 고려 사항은 qtree 가져오기에도 적용됩니다.

tridentctl을 사용하여 **qtree** 가져오기

qtree를 flexvol/qtree 형식의 볼륨 이름으로 지정합니다. 다음 예에서는 백엔드 `ontap_nas_economy`의 FlexVol 볼륨 `flexvol1`에서 qtree `qtree1`를 가져옵니다.

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-pvc-file>
```

qtree의 이름을 변경하지 않고 가져오려면 --no-rename 플래그를 추가합니다. Trident가 수명 주기를 관리하지 않고 qtree를 가져오려면 --no-manage 플래그를 추가합니다.

PVC 어노테이션을 사용하여 **qtree** 가져오기

다음 주석을 사용하여 PVC를 생성하십시오.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
```

```
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

기존 FlexVol 볼륨 및 qtree 이름을 유지하려면 `trident.netapp.io/importNoRename: "true"` 어노테이션을 추가하십시오. qtree를 관리되지 않는 형식으로 가져오려면 `trident.netapp.io/notManaged: "true"` 어노테이션을 추가하십시오.

명명 및 관리 동작

관리형 가져오기(기본값)

Trident는 qtree의 이름을 PVC 이름과 일치하도록 변경하고 FlexVol 볼륨의 이름을 `ontap-nas-economy` 명명 규칙으로 변경합니다. 접합 경로는 변경되지 않습니다.

`--no-rename`

Trident는 기존 FlexVol 볼륨 및 qtree 이름을 유지합니다. 이름은 이미 Trident 명명 규칙을 따라야 하며, 그렇지 않으면 Trident가 qtree에 대한 삭제 또는 마운트 작업을 관리할 수 없습니다.

`--no-manage`

Trident는 qtree를 참조하지만 해당 수명 주기를 관리하지는 않습니다. 이름은 Trident 명명 규칙을 따라야 하며, Trident는 qtree에 대한 자동 내보내기 정책을 사용하거나 관리하지 않습니다.

내보내기 정책 동작

백엔드에서 자동 내보내기 정책을 사용하는 경우, 가져온 qtree는 각 qtree에 고유한 내보내기 정책이 적용되는 qtree별 내보내기 정책 모델을 사용합니다. Trident는 가져오기 시 qtree에 빈 내보내기 정책을 할당합니다. 게시 시, 자동 내보내기가 활성화되어 있고 기존 FlexVol 내보내기 정책이 `default`이거나 비어 있는 경우 Trident는 FlexVol 볼륨을 백엔드별 내보내기 정책으로 업데이트합니다.

주의	FlexVol 볼륨 내보내기 정책을 업데이트하면 현재 마운트된 동일한 FlexVol 볼륨 내의 다른 qtree에 영향을 줄 수 있습니다. FlexVol 볼륨의 qtree가 마운트 해제된 경우에만 qtree를 가져오십시오.
경고	두 개의 Kubernetes 클러스터가 동일한 활성 SVM을 가리키도록 설정하지 마십시오. 또한 다른 클러스터에서 관리 중인 qtree를 가져오지 마십시오. 두 클러스터가 동일한 볼륨을 관리하는 경우, 충돌하는 작업으로 인해 데이터가 손상되거나 파괴될 수 있습니다.

ONTAP SAN

Trident 다음을 사용하여 볼륨 가져오기를 지원합니다. `ontap-san` (iSCSI, NVMe/TCP 및 FC) 및 `ontap-san-economy` 운전자.

Trident 단일 LUN을 포함하는 ONTAP SAN FlexVol 볼륨을 가져올 수 있습니다. 이는 다음과 일치합니다. `ontap-san` 각 PVC에 대한 FlexVol volume 과 FlexVol volume 내의 LUN을 생성하는 드라이버입니다. Trident FlexVol volume 가져와 PVC 정의와 연결합니다. Trident 수입이 가능합니다 `ontap-san-economy` 여러 LUN을 포함하는 볼륨.

다음 예는 관리형 볼륨과 비관리형 볼륨을 가져오는 방법을 보여줍니다.

관리 볼륨

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

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

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

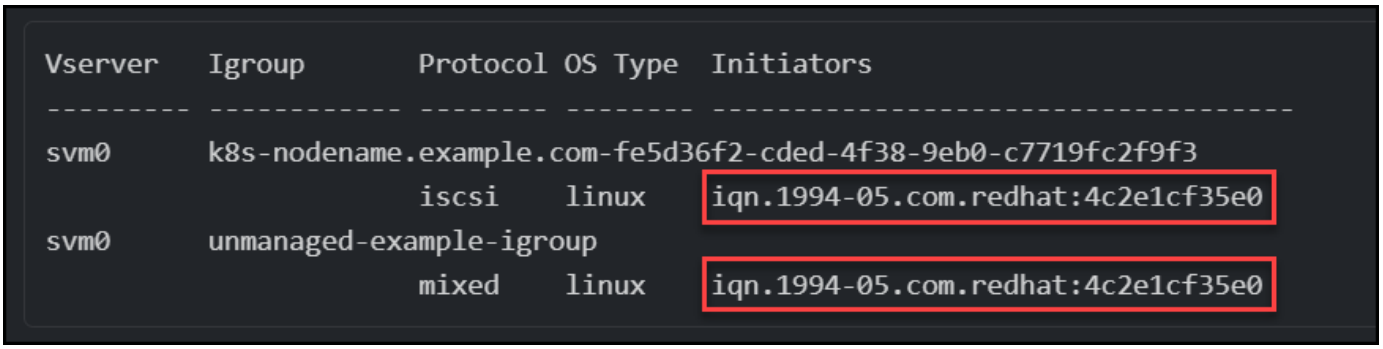
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

관리되지 않는 볼륨

다음 예에서는 를 가져옵니다 `unmanaged_example_volume` 를 누릅니다 `ontap_san` 백엔드:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

다음 예에 표시된 것처럼 IQN을 Kubernetes 노드 IQN과 공유하는 igroup에 LUN이 매핑되어 있는 경우 오류가 발생합니다. `LUN already mapped to initiator(s) in this group`. 볼륨을 가져오려면 이니시에이터를 제거하거나 LUN 매핑을 해제해야 합니다.



ONTAP SAN-economy 예

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

관리 볼륨

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

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

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

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

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

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

관리되지 않는 볼륨

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

FlexVol 명명 형식

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX`는 백엔드 구성의 `storagePrefix` 값입니다. 기본값은 `trident`입니다.

- `RANDOMSTRING`은(는) 선택한 문자열입니다.

LUN 명명 요구 사항

LUN의 이름은 `lun0`이어야 합니다.

예

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

```
trident_lun_pool_xyz_randomstring/lun0
```

요소

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

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

다음 예제에서는 을 가져옵니다 `element-managed` 백엔드의 볼륨 `element_default`.

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

+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
|                   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 드라이버를 사용하여 볼륨 가져오기를 `azure-netapp-files` 지원합니다.

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

다음 예제에서는 을 가져옵니다 `azure-netapp-files` 백엔드의 볼륨 `azurenetappfiles_40517` 볼륨 경로

포함 `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 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 볼륨

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

다음 예제는 backend `backend-tbc-gcnv1`에서 volume `testvoleasiaeast1`을 가져옵니다.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

다음 예에서는 동일한 영역에 두 개의 볼륨이 있을 때 볼륨을 가져옵니다 `google-cloud-netapp-volumes`.

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
```

```
-f <path-to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Trident 볼륨 이름 및 레이블 사용자 지정

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

시작하기 전에

사용자 지정 가능한 볼륨 이름 및 레이블 지원:

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

제한 사항

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

사용자 지정 가능한 볼륨 이름의 주요 동작

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

이름 템플릿 및 레이블이 있는 백엔드 구성 예

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

루트 수준의 예

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

풀 레벨 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 }}"
      }
    }
  ]
}
```

템플릿 예제를 명명합니다

- 예 1 *:

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

- 예 2 *:

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

고려해야 할 사항

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

여러 네임스페이스에서 **Trident NFS** 볼륨 공유

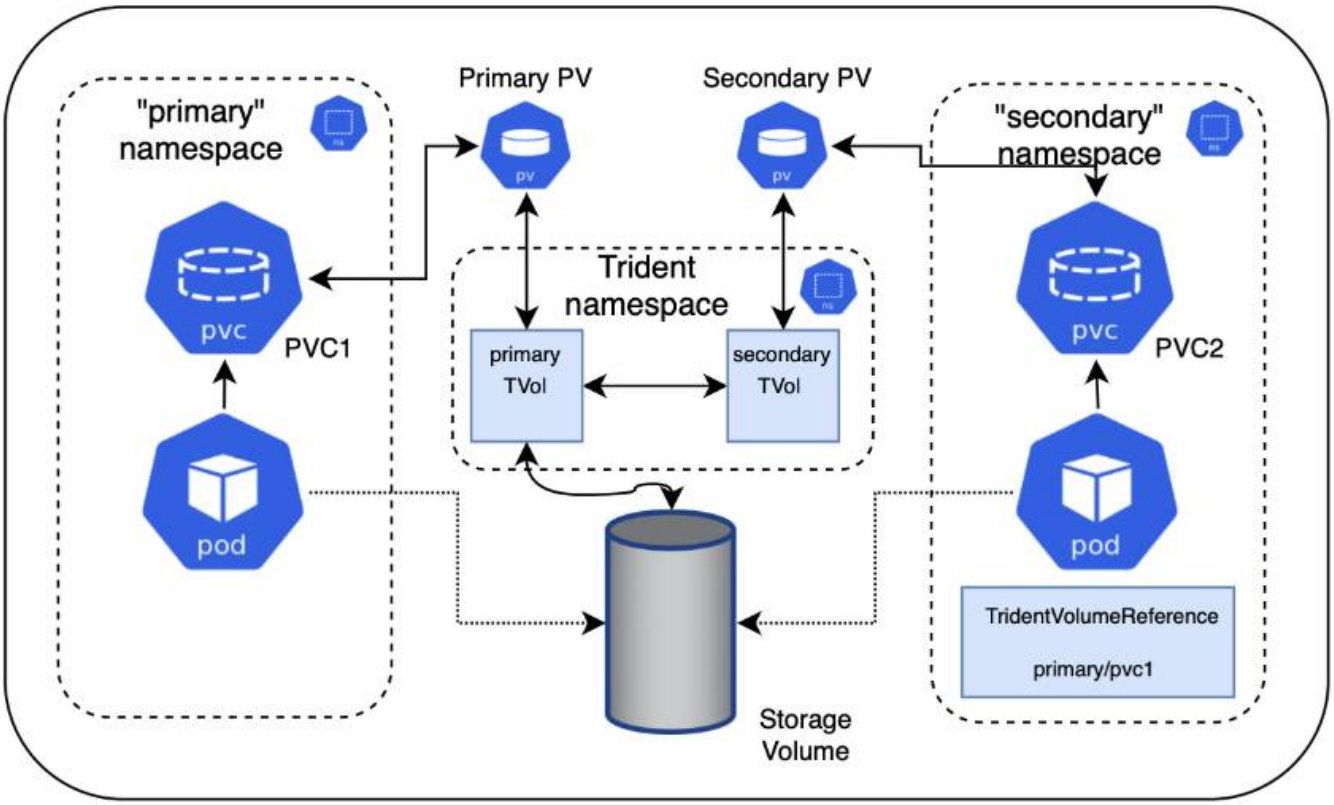
Trident를 사용하면 기본 네임스페이스에 볼륨을 생성한 후 하나 이상의 보조 네임스페이스에서 공유할 수 있습니다.

피처

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

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

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



빠른 시작

몇 단계만으로 NFS 볼륨 공유를 설정할 수 있습니다.

1 볼륨을 공유하도록 소스 **PVC**를 구성합니다

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

2 대상 네임스페이스에서 **CR**을 만들 수 있는 권한을 부여합니다

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

3 대상 네임스페이스에서 트리젠**VolumeReference** 를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 트리엔VolumeReference CR을 생성합니다.

4 대상 네임스페이스에서 하위 **PVC**를 만듭니다

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

소스 및 대상 네임스페이스를 구성합니다

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

단계

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

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

Trident은 PV와 백엔드 NFS 스토리지 볼륨을 생성합니다.

참고

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

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하세요.

3. * 대상 네임스페이스 소유자: * 소스 네임스페이스를 참조하는 대상 네임스페이스에서 트리젠VolumeReference CR을 만듭니다 `pvc1`.

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

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

```
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
```

참고	대상 PVC의 크기는 소스 PVC보다 작거나 같아야 합니다.

결과

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

공유 볼륨을 삭제합니다

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

사용 `tridentctl get` 하위 볼륨을 쿼리합니다

[`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`: 하위 원본 볼륨으로 쿼리를 제한합니다.
- `--subordinateOf string`: 볼륨 부하로 쿼리 제한.

제한 사항

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

를 참조하십시오

네임스페이스 간 볼륨 액세스에 대한 자세한 내용은 다음을 참조하십시오.

- 데모를 시청해보시기 "NetAppTV를 참조하십시오"바랍니다.

네임스페이스 간 **Trident** 볼륨 복제

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

필수 구성 요소

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

참고	네임스페이스 간 복제는 다음에 대해서만 지원됩니다. `ontap-san` 그리고 `ontap-nas` 스토리지 드라이버. 읽기 전용 복제본은 지원되지 않습니다.

빠른 시작

몇 단계만 거치면 볼륨 복제를 설정할 수 있습니다.

1. 볼륨을 복제하도록 소스 **PVC**를 구성합니다

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

2 대상 네임스페이스에서 **CR**을 만들 수 있는 권한을 부여합니다

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

3 대상 네임스페이스에서 트리젠**VolumeReference** 를 생성합니다

대상 네임스페이스의 소유자는 소스 PVC를 참조하기 위해 트리엔VolumeReference CR을 생성합니다.

4 대상 네임스페이스에 클론 **PVC**를 생성합니다

대상 네임스페이스의 소유자는 PVC를 생성하여 소스 네임스페이스에서 PVC를 복제합니다.

소스 및 대상 네임스페이스를 구성합니다

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

단계

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

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

Trident은 PV와 백엔드 스토리지 볼륨을 생성합니다.

참고

- 쉼표로 구분된 목록을 사용하여 PVC를 여러 네임스페이스에 공유할 수 있습니다. `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`예를 들어,
- 을 사용하여 모든 네임스페이스에 공유할 수 * 있습니다. 예를 들면, 다음과 같습니다. `trident.netapp.io/cloneToNamespace: *`

◦ 주석을 포함하도록 PVC를 업데이트할 수 `cloneToNamespace` 있습니다.

2. 클러스터 관리자: 대상 네임스페이스 소유자에게 대상 네임스페이스에서 TridentVolumeReference CR을 생성할 수 있는 권한을 부여하기 위해 적절한 RBAC가 있는지 확인하십시오.(`namespace2`).
3. 대상 네임스페이스 소유자: 대상 네임스페이스에 소스 PVC `pvc1`를 참조하는 TridentVolumeReference CR을 생성합니다.

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

4. * 대상 네임스페이스 소유자: * (`pvc2`대상 네임스페이스에서 PVC 생성(`namespace2`) 또는 `cloneFromSnapshot`, `cloneFromNamespace` 주석을 사용하여 `cloneFromPVC` 소스 PVC를 지정합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

제한 사항

- ONTAP-NAS-이코노미 드라이버를 사용하여 프로비저닝된 PVC의 경우 읽기 전용 클론은 지원되지 않습니다.

SnapMirror를 사용하여 볼륨을 복제합니다

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

사용하여 다음 작업을 수행할 수 있습니다.

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

복제 사전 요구 사항

시작하기 전에 다음과 같은 사전 요구 사항이 충족되는지 확인하십시오.

ONTAP 클러스터

- * Trident *: Trident 버전 22.10 이상이 ONTAP를 백엔드로 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 있어야 합니다.
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 자세한 내용은 을 "ONTAP의 SnapMirror 라이센스 개요" 참조하십시오.

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

참고	SnapMirror 비동기 보호만 지원됩니다.

피어링

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

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

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

지원되는 드라이버

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

참고	SnapMirror를 사용한 볼륨 복제는 ASA r2 시스템에서는 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 다음을 참조하세요. "ASA R2 스토리지 시스템에 대해 알아보십시오" .

대칭 복사된 **PVC**를 작성합니다

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

단계

1. 운영 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. 매개 변수를 사용하여 StorageClass 개체를 `trident.netapp.io/replication: true` 만듭니다.

예

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. 이전에 생성된 StorageClass를 사용하여 PVC를 생성합니다.

예

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

c. 로컬 정보로 MirrorRelationship CR을 생성합니다.

예

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

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

d. TridentMirrorRelationship CR을 가져와 PVC의 내부 이름과 SVM을 얻습니다.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. 보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.
   a. trident.netapp.io/replication: true 매개 변수를 사용하여 StorageClass 를 만듭니다.

      예

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. 대상 및 소스 정보를 사용하여 MirrorRelationship CR을 생성합니다.

      예

      ```
      kind: TridentMirrorRelationship
      apiVersion: trident.netapp.io/v1
      metadata:
        name: csi-nas
      spec:
        state: established
        replicationPolicy: MirrorAllSnapshots
      ```

```
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

TMR `spec`에는 4개의 최상위 필드가 있습니다. 원하는 관계 상태를 설정하는 `state`(볼륨 복제 상태입니다 참조), 로컬 PVC를 원격 볼륨 핸들과 연결하는 `volumeMappings`, 그리고 아래에 설명된 선택적 `replicationPolicy` 및 `replicationSchedule` 필드입니다.

`replicationPolicy` 및 `replicationSchedule` 필드는 선택 사항입니다. `replicationPolicy`를 ONTAP의 비동기식 `SnapMirror` 정책 이름으로 설정하고 `replicationSchedule`를 ONTAP 스케줄 이름으로 설정하십시오. 두 필드 중 하나라도 생략하면 `Trident`는 백엔드에 구성된 해당 복제 정책 또는 스케줄을 사용합니다. 관계를 생성할 때 이러한 필드를 설정하십시오. 관계가 설정된 후에는 정책이나 스케줄을 변경할 수 없습니다. 다른 설정을 사용하려면 `TMR`을 삭제하고 다시 생성하십시오.

Trident는 구성된 복제 정책 및 일정을 사용하여 SnapMirror 관계를 생성하고 초기화합니다.

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

   예

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

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

볼륨 복제 상태입니다

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

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

비계획 페일오버 중에 보조 **PVC**를 승격합니다

보조 Kubernetes 클러스터에서 다음 단계를 수행합니다.

- TridentMirrorRelationship의 _spec.state_field를 로 `promoted`업데이트합니다.

계획된 페일오버 중에 보조 **PVC**를 승격합니다

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

단계

1. 운영 Kubernetes 클러스터에서 PVC의 스냅샷을 생성하고 스냅샷이 생성될 때까지 기다립니다.
2. 기본 Kubernetes 클러스터에서 TridentTridentSnapshotInfoCR을 생성하여 내부 정보를 얻으십시오.

   예

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

   a. 보조 Kubernetes 클러스터에서 *TridentMirrorRelationship* CR의 *spec.state* 필드를 _promoted_로 업데이트하고 _spec.volumeMappings[].promotedSnapshotHandle_을 스냅샷의 internalName으로 설정합니다.
3. 보조 Kubernetes 클러스터에서 승격될 TridentMirrorRelationship의 상태(status.state 필드)를 확인합니다.

페일오버 후 미러 관계를 복구합니다

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

단계

1. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 _spec.remoteVolumeHandle_field 값이 업데이트되었는지 확인합니다.
2. 보조 Kubernetes 클러스터에서 TridentMirrorRelationship의 *spec.state* 필드를 `reestablished`로 업데이트합니다.

추가 작업

Trident는 1차 볼륨과 2차 볼륨에서 다음 작업을 지원합니다.

1차 PVC를 새로운 2차 PVC로 복제합니다

이미 1차 PVC와 2차 PVC가 있는지 확인하십시오.

단계

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

대칭 복사, 1차 또는 2차 PVC의 크기를 조정합니다

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

PVC에서 복제를 제거합니다

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

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

PVC 삭제(이전에 미러링됨)

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

TMR을 삭제합니다

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

ONTAP가 온라인 상태일 때 미러 관계를 업데이트합니다

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

ONTAP이 오프라인일 때 미러 관계를 업데이트합니다

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

예

```
apiVersion: trident.netapp.io/v1
```

```
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` TridentActionMirrorUpdate CRD의 상태를 반영합니다. 이 값은 *SUCCEEDED*,*In Progress* 또는_Failed_에서 가져올 수 있습니다.

CSI 토폴로지를 사용합니다

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

개요

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

> 팁 CSI 토폴로지 기능에 대해 자세히 알아보십시오 "여기".

Kubernetes는 두 가지 고유한 볼륨 바인딩 모드를 제공합니다.

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

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

필요한 것

CSI 토폴로지를 사용하려면 다음이 필요합니다.

- 를 실행하는 Kubernetes 클러스터 "지원되는 Kubernetes 버전"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"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 레이블(`topology.kubernetes.io/region` 및 `topology.kubernetes.io/zone`)이 있어야 합니다. Trident에서 토폴로지 인식 프로비저닝을 위해서는 `topology.kubernetes.io/region` 레이블이 필수입니다. 이러한 레이블은 Trident를 설치하기 전에 노드에 있어야 합니다. 설치 후 토폴로지 레이블을 추가하거나 변경하는 경우, Trident 노드 Pod를 다시 시작하면 Trident가 변경 사항을 인식할 수 있습니다.

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

다음은 백엔드 정의의 예입니다.

YAML

```
---
```

```
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON을 참조하십시오

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

참고

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

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

```
---
version: 1
storageDriverName: ontap-nas
```

```
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

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

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

클러스터의 노드에 제공되는 토폴로지 레이블을 기반으로 StorageClasses를 정의하여 토폴로지 정보를 포함할 수 있습니다. 이렇게 하면 PVC 요청에 대한 후보 역할을 하는 스토리지 풀과 Trident에서 제공하는 볼륨을 사용할 수 있는 노드의 하위 세트가 결정됩니다.

다음 예를 참조하십시오.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
```

```
      values:
        - us-east1
parameters:
  fsType: ext4
```

위에 제공된 StorageClass 정의에서 `volumeBindingMode` 는 로 `WaitForFirstConsumer`설정됩니다. 이 `StorageClass`에 요청된 `PVC`는 `POD`에서 참조될 때까지 작동하지 않습니다. 및 는 `allowedTopologies` 사용할 영역 및 영역을 제공합니다. `netapp-san-us-east1`StorageClass는 위에 정의된 백엔드에 `PVC`를 `san-backend-us-east1` 생성합니다.

3단계: **PVC** 생성 및 사용

StorageClass가 생성되어 백엔드에 매핑되면 PVC를 생성할 수 있습니다.

아래의 '샘플'을 참조하십시오.

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

이 매니페스트를 사용하여 PVC를 만들면 다음과 같은 결과가 발생합니다.

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
```

```
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Trident에서 볼륨을 생성하여 PVC에 바인딩하려면 POD에서 PVC를 사용합니다. 다음 예를 참조하십시오.

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
```

```
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

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

다음 출력을 참조하십시오.

```
kubectl get pods -o wide
NAME         READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1    1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME       STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san    Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

포함할 백엔드를 업데이트합니다 **supportedTopologies**

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

자세한 내용을 확인하십시오

- "컨테이너에 대한 리소스를 관리합니다"
- "노드 선택기"
- "친화성 및 반친화성"
- "오염과 내약입니다"

Trident 스냅샷 작업

영구 볼륨(PVS)의 Kubernetes 볼륨 스냅샷은 볼륨의 시점 복사본을 지원합니다. Trident를 사용하여 생성된 볼륨의 스냅샷을 생성하고, Trident 외부에서 생성된 스냅샷을 가져오고, 기존 스냅샷에서 새 볼륨을 생성하고, 스냅샷에서 볼륨 데이터를 복구할 수 있습니다.

개요

볼륨 스냅샷은 다음에서 지원됩니다. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `azure-netapp-files` , 그리고 `google-cloud-netapp-volumes` 운전자.

시작하기 전에

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

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

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

볼륨 스냅샷을 생성합니다

단계

1. 을 생성합니다 `VolumeSnapshotClass`. 자세한 내용은 을 참조하십시오 "VolumeSnapshotClass".
   - 가 `driver` Trident CSI 드라이버를 가리킵니다.
   - `deletionPolicy` 있을 수 있습니다 `Delete` 또는 `Retain`. 를 로 설정한 경우 `Retain`, 스토리지 클러스터의 기본 물리적 스냅샷은 가 있는 경우에도 유지됩니다 `VolumeSnapshot` 객체가 삭제되었습니다.

   예

   ```
   cat snap-sc.yaml
   ```

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

2. 기존 PVC의 스냅샷을 생성합니다.

   예
   - 이 예에서는 기존 PVC의 스냅샷을 생성합니다.

   ```
   cat snap.yaml
   ```

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
   ```

```
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

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

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

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

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

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

볼륨 스냅샷에서 **PVC**를 생성합니다

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

경고 PVC는 소스 볼륨과 동일한 백엔드에서 생성됩니다. 을 참조하십시오 "KB: Trident PVC 스냅샷에서 PVC를 생성하는 것은 대체 백엔드에서 생성할 수 없습니다".

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

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

볼륨 스냅샷을 가져옵니다

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

시작하기 전에

Trident에서 스냅샷의 상위 볼륨을 생성하거나 가져와야 합니다.

단계

1. * 클러스터 관리자: * `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 객체를 생성합니다. 그러면 Trident에서 스냅샷 워크플로우가 시작됩니다.
   - 에서 백엔드 스냅샷의 이름을 지정합니다 `annotations` 현재 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`에서 `snapshotHandle`지정합니다. 이 정보는 통화에서 외부 스냅숏이 `Trident`에 제공하는 유일한 `ListSnapshots` 정보입니다.

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

   예

다음 예제에서는 을 만듭니다 `VolumeSnapshotContent` 백엔드 스냅샷을 참조하는 객체입니다 `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. * 클러스터 관리자: * 를 생성합니다 `VolumeSnapshot` 을 참조하는 CR `VolumeSnapshotContent` 오브젝트. 그러면 를 사용할 수 있는 액세스가 필요합니다 `VolumeSnapshot` 지정된 네임스페이스에서.

   예

   다음 예제에서는 을 만듭니다 `VolumeSnapshot` CR 이름 `import-snap` 을 참조합니다 `VolumeSnapshotContent` 이름 지정 `import-snap-content`.

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

3. * 내부 처리(조치 필요 없음): * 외부 스냅숏 작성자가 새로 생성된 을 `VolumeSnapshotContent` 인식하고 `ListSnapshots` 통화를 실행합니다. Trident가 를 `TridentSnapshot`생성합니다.
   - 외부 스냅숏 작성기가 를 설정합니다 `VolumeSnapshotContent` 를 선택합니다 `readyToUse` 및 `VolumeSnapshot` 를 선택합니다 `true`.
   - Trident가 돌아왔습니다 `readyToUse=true`.
4. * 모든 사용자: * 를 생성합니다 `PersistentVolumeClaim` 를 눌러 새 를 참조합니다 `VolumeSnapshot`, 위치

`spec.dataSource` (또는 `spec.dataSourceRef`) name 은 입니다 `VolumeSnapshot` 이름.

예

다음 예에서는 를 참조하는 PVC를 작성합니다 `VolumeSnapshot` 이름 지정 `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

스냅샷을 사용하여 볼륨 데이터를 복구합니다

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

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅숏에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

참고 | 스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 대한 변경 사항은 손실됩니다.

스냅샷에서 전체 볼륨 복원

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

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

시작하기 전에

바인딩된 PVC 및 사용 가능한 볼륨 스냅샷이 있어야 합니다.

- PVC 상태가 Bound인지 확인한다.

```
kubectl get pvc
```

- 볼륨 스냅샷을 사용할 준비가 되었는지 확인합니다.

```
kubectl get vs
```

단계

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

   참고 | TASR CR은 PVC & VS가 있는 네임스페이스에 있어야 합니다.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. CR을 적용하여 스냅샷에서 복원합니다. 이 예는 스냅샷에서 `pvc1`복구합니다.

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

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

결과

Trident는 스냅샷에서 데이터를 복원합니다. 스냅샷 복구 상태를 확인할 수 있습니다.

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
```

```
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

참고

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

연결된 스냅샷이 있는 **PV**를 삭제합니다

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

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포 시 스냅샷 컨트롤러와 CRD가 포함되지 않은 경우 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
```

```
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.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
```

참고	필요한 경우 를 엽니다 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 및 업데이트 `namespace` 네임스페이스로.

관련 링크

- "볼륨 스냅숏"
- "VolumeSnapshotClass"

Trident 볼륨 그룹 스냅샷 작업

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

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

볼륨 그룹 스냅샷 생성

볼륨 그룹 스냅샷은 다음 스토리지 드라이버에서 지원됩니다.

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

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

시작하기 전에

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

  Kubernetes 배포판에 외부 스냅샷 컨트롤러 및 CRD가 포함되어 있지 않은 경우 다음을 참조하세요. 볼륨 스냅샷 컨트롤러를 배포합니다 .

  참고 | GKE 환경에서 주문형 볼륨 그룹 스냅샷을 생성하는 경우 스냅샷 컨트롤러를 생성하지 마세요. GKE는 내장된 숨겨진 스냅샷 컨트롤러를 사용합니다.

- 스냅샷 컨트롤러 YAML에서 다음을 설정합니다. `CSIVolumeGroupSnapshot` 볼륨 그룹 스냅샷이 활성화되도록 기능 게이트를 'true'로 설정합니다.
- 볼륨 그룹 스냅샷을 생성하기 전에 필요한 볼륨 그룹 스냅샷 클래스를 생성합니다.
- VolumeGroupSnapshot을 생성하려면 모든 PVC/볼륨이 동일한 SVM에 있는지 확인하세요.

단계

- VolumeGroupSnapshot을 생성하기 전에 VolumeGroupSnapshotClass를 생성하세요. 자세한 내용은 을 "볼륨 그룹 스냅샷 클래스"참조하십시오.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 기존 저장 클래스를 사용하여 필요한 라벨이 있는 PVC를 만들거나, 이러한 라벨을 기존 PVC에 추가합니다.

  다음 예제에서는 다음을 사용하여 PVC를 생성합니다. `pvc1-group-snap` 데이터 소스 및 레이블로 `consistentGroupSnapshot: groupA` . 요구 사항에 따라 레이블 키와 값을 정의합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- 동일한 레이블로 VolumeGroupSnapshot을 만듭니다. (`consistentGroupSnapshot: groupA`) PVC에 지정됨.

  이 예제에서는 볼륨 그룹 스냅샷을 생성합니다.

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

그룹 스냅샷을 사용하여 볼륨 데이터 복구

볼륨 그룹 스냅샷의 일부로 생성된 개별 스냅샷을 사용하여 개별 영구 볼륨을 복원할 수 있습니다. 볼륨 그룹 스냅샷을 하나의 단위로 복구할 수는 없습니다.

볼륨 스냅샷 복원 ONTAP CLI를 사용하여 볼륨을 이전 스냅숏에 기록된 상태로 복원합니다.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

참고	스냅샷 복사본을 복원하면 기존 볼륨 구성이 덮어쓰여집니다. 스냅샷 복사본이 생성된 후 볼륨 데이터에 대한 변경 사항은 손실됩니다.

스냅샷에서 전체 볼륨 복원

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

자세한 내용은 을 "스냅샷에서 전체 볼륨 복원"참조하십시오.

연관된 그룹 스냅샷이 있는 **PV** 삭제

그룹 볼륨 스냅샷을 삭제할 때:

- 그룹의 개별 스냅샷이 아닌 VolumeGroupSnapshots 전체를 삭제할 수 있습니다.
- 스냅샷이 있는 동안 해당 PersistentVolume이 삭제되면 Trident는 해당 볼륨을 "삭제 중" 상태로 전환합니다. 볼륨을 안전하게 제거하기 전에 스냅샷을 제거해야 하기 때문입니다.
- 그룹화된 스냅샷을 사용하여 복제본을 만든 다음 그룹을 삭제하려는 경우 복제본 분할 작업이 시작되고 분할이 완료될 때까지 그룹을 삭제할 수 없습니다.

볼륨 스냅샷 컨트롤러를 배포합니다

Kubernetes 배포 시 스냅샷 컨트롤러와 CRD가 포함되지 않은 경우 다음과 같이 배포할 수 있습니다.

단계

1. 볼륨 스냅샷 CRD를 생성합니다.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_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
```

참고	필요한 경우 를 엽니다 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` 및 업데이트 `namespace` 네임스페이스로.

관련 링크

- "볼륨 그룹 스냅샷 클래스"
- "볼륨 스냅숏"

Trident 관리 및 모니터링

Trident를 업그레이드합니다

Trident를 업그레이드합니다

24.02 릴리즈부터 Trident는 4개월 릴리즈 주기를 따르며 매년 3개의 주요 릴리즈를 제공합니다. 각각의 새 릴리스는 이전 릴리스에서 빌드되며 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 릴리즈로 업그레이드할 때 다음 사항을 고려하십시오.

- 특정 Kubernetes 클러스터의 모든 네임스페이스에 Trident 인스턴스는 하나만 설치해야 합니다.
- Trident 23.07 이상에서는 v1 볼륨 스냅샷이 필요하며 알파 또는 베타 스냅샷을 더 이상 지원하지 않습니다.
- 업그레이드할 때 Trident에서 사용되는 를 `StorageClasses` 제공하는 것이 중요합니다 `parameter.fsType`. 기존 볼륨을 중단하지 않고 삭제 및 재생성할 수 `StorageClasses` 있습니다.
  - 이것은 시행에 대한 ** 요구 사항입니다 "보안 컨텍스트" SAN 볼륨:
  - sample input 디렉토리에는 https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.templ와 같은 예가 포함되어 있습니다[`storage-class-basic.yaml.templ`^] 및 링크: `storage-class-bronze-default.yaml`.
  - 자세한 내용은 을 참조하십시오 "알려진 문제".

1단계**:** 버전을 선택합니다

Trident 버전은 날짜 기반 `YY.MM` 명명 규칙을 따릅니다. 여기서 "YY"는 연도의 마지막 두 자리이고 "MM"은 월입니다. DOT 릴리스는 `YY.MM.X` 규칙을 따릅니다. 여기서 "X"는 패치 수준입니다. 업그레이드할 버전에 따라 업그레이드할 버전을 선택합니다.

- 설치된 버전의 4 릴리스 창 내에 있는 모든 대상 릴리스에 대해 직접 업그레이드를 수행할 수 있습니다. 예를 들어 24.06(또는 24.06 도트 릴리스)에서 25.06로 직접 업그레이드할 수 있습니다.
- 릴리스 4개가 아닌 다른 릴리스에서 업그레이드하는 경우 여러 단계로 업그레이드를 수행합니다. 에서 4개의 릴리스 윈도우에 맞는 가장 최신 릴리스로 업그레이드하려는 의 업그레이드 지침을 "이전 버전" 사용합니다. 예를 들어, 23.07을 실행하고 있고 25.06로 업그레이드하려는 경우:
  a. 23.07에서 24.06로 첫 번째 업그레이드.
  b. 그런 다음 24.06에서 25.06로 업그레이드합니다.

2단계**:** 원래 설치 방법을 결정합니다

Trident를 처음 설치할 때 사용한 버전을 확인하려면 다음을 수행합니다.

1. 사용 `kubectl get pods -n trident` 를 눌러 포드를 검사합니다.

- 운영자 포드가 없으면 를 사용하여 Trident를 설치한 `tridentctl` 것입니다.
- 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.

2. 운영자 포드가 있는 경우 를 `kubectl describe torc` 사용하여 Trident가 Helm을 사용하여 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Helm을 사용하여 Trident를 설치한 것입니다.
   - Helm 레이블이 없는 경우 Trident 연산자를 사용하여 Trident를 수동으로 설치했습니다.

3단계**:** 업그레이드 방법을 선택합니다

일반적으로 초기 설치에 사용했던 것과 동일한 방법으로 업그레이드하는 것이 좋지만, "설치 방법 간에 이동합니다". Trident를 업그레이드하는 두 가지 옵션이 있습니다.

- "Trident 연산자를 사용하여 업그레이드합니다"

팁	검토할 것을 제안합니다 "운영자 업그레이드 워크플로우를 이해합니다" 작업자와 업그레이드하기 전에

*

운영자와 함께 업그레이드하십시오

Trident 오퍼레이터 업그레이드 워크플로 이해

Trident 연산자를 사용하여 Trident를 업그레이드하기 전에 업그레이드 중에 발생하는 백그라운드 프로세스를 이해해야 합니다. 여기에는 Trident 컨트롤러, 컨트롤러 Pod 및 노드 Pod, 롤링 업데이트를 사용하는 노드 DemonSet의 변경 사항이 포함됩니다.

Trident 운영자 업그레이드 처리

Trident를 설치 및 업그레이드하는 많은 "Trident 연산자를 사용할 때의 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 컨트롤러 배포 및 노드 DemonSet을 삭제하고 다시 만듭니다.
- Trident 컨트롤러 Pod 및 Trident 노드 Pod를 새로운 버전으로 교체합니다.
  - 노드가 업데이트되지 않으면 나머지 노드가 업데이트됩니다.
  - 실행 중인 Trident 노드 Pod가 있는 노드에서만 볼륨을 마운트할 수 있습니다.

팁	Kubernetes 클러스터의 Trident 아키텍처에 대한 자세한 내용은 을 참조하십시오"Trident 아키텍처".

운영자 업그레이드 워크플로우

Trident 연산자를 사용하여 업그레이드를 시작하는 경우:

1. Trident 운영자 *:

a. 현재 설치된 Trident 버전(version_n_)을 검색합니다.

b. CRD, RBAC, Trident SVC를 포함한 모든 Kubernetes 오브젝트를 업데이트합니다.

c. 버전_n_에 대한 Trident 컨트롤러 배포를 삭제합니다.

d. 버전_n+1_에 대한 Trident 컨트롤러 배포를 생성합니다.

2. * Kubernetes * 는 _n+1_용 Trident 컨트롤러 포드를 생성합니다.

3. Trident 운영자 *:

a. _n_에 대한 Trident 노드 데모 세트를 삭제합니다. 운용자는 Node Pod 종료를 기다리지 않는다.

b. _n+1_에 대한 Trident 노드 데모 세트를 생성합니다.

4. * Kubernetes * 는 Trident 노드 Pod_n_을(를) 실행하지 않는 노드에서 Trident 노드 Pod를 생성합니다. 따라서 노드에 여러 버전의 Trident 노드 Pod가 둘 이상 있지 않도록 합니다.

Trident operator 또는 Helm을 사용하여 Trident 설치를 업그레이드합니다

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 선택합니다"를 검토하십시오.

수동 설치를 업그레이드합니다

클러스터 범위 Trident Operator 설치에서 다른 클러스터 범위 Trident Operator 설치로 업그레이드할 수 있습니다. 모든 Trident 버전은 클러스터 범위 연산자를 사용합니다.

참고	네임스페이스 범위 연산자(버전 20.07 ~ 20.10)를 사용하여 설치된 Trident에서 업그레이드하려면 Trident의 업그레이드 지침을 "설치된 버전"사용하십시오.

이 작업에 대해

Trident는 운영자를 설치하고 Kubernetes 버전에 대한 관련 오브젝트를 생성하는 데 사용할 수 있는 번들 파일을 제공합니다.

- Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 "Bundle_post_1_25.YAML"사용합니다.

시작하기 전에

실행 중인 Kubernetes 클러스터를 사용하고 있는지 확인합니다 "지원되는 Kubernetes 버전".

단계

1. Trident 버전 확인:

```
./tridentctl -n trident version
```

2. 업데이트 `operator.yaml` , `tridentorchestrator_cr.yaml` , 그리고 `post_1_25_bundle.yaml` 업그레이드하려는 버전(예: 25.06)에 대한 레지스트리와 이미지 경로, 그리고 올바른 비밀번호를 사용합니다.

3. 현재 Trident 인스턴스를 설치하는 데 사용된 Trident 연산자를 삭제합니다. 예를 들어, 25.02에서 업그레이드하는 경우 다음 명령을 실행합니다.

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 를 사용하여 초기 설치를 사용자 지정한 경우 `TridentOrchestrator` 속성을 편집할 수 있습니다 `TridentOrchestrator` 설치 매개 변수를 수정하는 개체입니다. 여기에는 오프라인 모드에 대해 미러링된 Trident 및 CSI 이미지 레지스트리를 지정하는 변경 사항, 디버그 로그 활성화 또는 이미지 풀 비밀을 지정하는 변경 사항이 포함될 수 있습니다.
5. _<bundle.yaml>_이 있는 환경에 맞는 올바른 번들 YAML 파일을 사용하여 Trident 설치하세요. `bundle_pre_1_25.yaml` 또는 `bundle_post_1_25.yaml` Kubernetes 버전에 따라 다릅니다. 예를 들어, Trident 25.06.0을 설치하는 경우 다음 명령을 실행합니다.

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. 트라이던트 토크를 편집하여 이미지 25.06.0을 포함시킵니다.

Helm 설치를 업그레이드합니다

Trident Helm 설치를 업그레이드할 수 있습니다.

경고	Trident가 설치된 Kubernetes 클러스터를 1.24에서 1.25 이상으로 업그레이드할 `true` `helm upgrade` 경우 클러스터를 업그레이드하기 전에 value.yaml을 명령으로 설정하거나 `--set excludePodSecurityPolicy=true` 명령에 추가해야 `excludePodSecurityPolicy` 합니다.

Trident Helm을 업그레이드하지 않고 이미 Kubernetes 클러스터를 1.24에서 1.25로 업그레이드한 경우에는 Helm 업그레이드가 실패합니다. Helm 업그레이드를 진행하려면 다음 단계를 전제 조건으로 수행하십시오.

1. 에서 helm-mapkubeapis 플러그인을 https://github.com/helm/helm-mapkubeapis 설치합니다.
2. Trident가 설치된 네임스페이스에서 Trident 릴리스에 대해 건식 실행을 수행합니다. 정리할 리소스가 나열됩니다.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. 정리 작업을 수행하려면 Helm을 사용하여 전체 실행을 수행합니다.

```
helm mapkubeapis trident  --namespace trident
```

단계

1. "Helm을 사용하여 Trident를 설치했습니다"한 경우 `helm upgrade trident netapp-trident/trident-operator --version 100.2610.0`를 사용하여 한 단계로 업그레이드할 수 있습니다. Helm 리포지토리를 추가하지 않았거나 업그레이드에 사용할 수 없는 경우:
   a. 에서 최신 Trident 릴리스를 "GitHub의 _Assets_섹션"다운로드하십시오.
   b. `helm upgrade` 명령을 사용합니다. 여기서 `trident-operator-26.10.1.tgz`는 업그레이드할 버전을 나타냅니다.

```
helm upgrade <name> trident-operator-26.10.1.tgz
```

참고	초기 설치 중에 사용자 정의 옵션을 설정하는 경우(예: Trident 및 CSI 이미지에 대한 전용 미러링된 레지스트리 지정) 를 추가합니다 `helm upgrade` 명령을 사용합니다 `--set` 이러한 옵션이 업그레이드 명령에 포함되도록 하려면 값이 기본값으로 재설정됩니다.

2. 실행 `helm list` 차트와 앱 버전이 모두 업그레이드되었는지 확인합니다. 실행 `tridentctl logs` 디버그 메시지를 검토합니다.

에서 업그레이드 `tridentctl` Trident 운영자에 설치

에서 Trident 운영자의 최신 릴리즈로 업그레이드할 수 있습니다 `tridentctl` 설치: 기존 백엔드 및 PVC를 자동으로 사용할 수 있습니다.

참고	설치 방법을 전환하기 전에 "설치 방법 간 이동"을(를) 검토하십시오.

단계

1. 최신 Trident 릴리스를 다운로드합니다.

```
# Download the release required [26.10.1]
mkdir 26.10.1
cd 26.10.1
wget
https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2. 매니페스트에서 트라이디오케스트레이터 CRD를 만듭니다.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 클러스터 범위 연산자를 같은 네임스페이스에 구현합니다.

```
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. `TridentOrchestrator`Trident 설치를 위한 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 확인합니다.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.10.0
```

tridentctl을 사용하여 **Trident** 업그레이드

를 사용하여 기존 Trident 설치를 쉽게 업그레이드할 수 `tridentctl` 있습니다.

이 작업에 대해

Trident 제거 및 재설치는 업그레이드로 작동합니다. Trident를 제거하면 Trident 배포에 사용되는 PVC(영구 볼륨 클레임) 및 PV(영구 볼륨)가 삭제되지 않습니다. 이미 프로비저닝된 PVS는 Trident가 오프라인 상태인 동안 사용 가능한 상태로 유지되며, Trident는 다시 온라인 상태가 된 후 그 사이에 생성된 모든 PVC에 대한 볼륨을 프로비저닝합니다.

시작하기 전에

업그레이드하기 전에 "업그레이드 방법을 선택합니다"를 검토한 후 `tridentctl`를 사용하십시오.

단계

1. 에서 uninstall 명령을 `tridentctl` 실행하여 CRD 및 관련 객체를 제외한 Trident와 연결된 모든 리소스를 제거합니다.

```
./tridentctl uninstall -n <namespace>
```

2. Trident를 다시 설치합니다. "tridentctl을 사용하여 Trident를 설치합니다"을 참조하십시오.

중요함	업그레이드 프로세스를 중단하지 마십시오. 설치 프로그램이 완료될 때까지 실행되는지 확인합니다.

tridentctl을 사용하여 Trident를 관리합니다

에는 "Trident 설치 프로그램 번들" Trident에 간단하게 액세스할 수 있는 명령줄 유틸리티가 포함되어 `tridentctl` 있습니다. 충분한 Privileges를 가진 Kubernetes 사용자는 이 툴을 사용하여 Trident를 설치하거나 Trident Pod가 포함된 네임스페이스를 관리할 수 있습니다.

명령 및 글로벌 플래그

실행할 수 있습니다 `tridentctl help` 에 사용할 수 있는 명령 목록을 가져옵니다 `tridentctl` 또는 를 추가합니다 `--help` 특정 명령에 대한 옵션 및 플래그 목록을 가져오려면 임의의 명령에 플래그를 지정합니다.

```
tridentctl [command] [--optional-flag]
```

Trident `tridentctl` 유틸리티는 다음 명령 및 글로벌 플래그를 지원합니다.

▼ 명령

`create`

Trident에 리소스를 추가합니다.

`delete`

Trident에서 하나 이상의 리소스를 제거합니다.

`get`

Trident에서 하나 이상의 리소스를 얻습니다.

`help`

모든 명령에 대한 도움말.

`images`

Trident에 필요한 컨테이너 이미지 표를 인쇄합니다.

import

기존 리소스를 Trident로 가져옵니다.

install

Trident를 설치합니다.

logs

Trident에서 로그를 인쇄합니다.

send

Trident에서 리소스를 보냅니다.

"제거"를 선택합니다

Trident를 제거합니다.

update

Trident에서 리소스를 수정합니다.

update backend state

백엔드 작업을 일시적으로 중단합니다.

upgrade

Trident에서 리소스를 업그레이드합니다.

'내전'

Trident 버전을 인쇄합니다.

▼ 글로벌 플래그

-d, --debug

디버그 출력.

-h, --help

도움말 `tridentctl`.

-k, --kubeconfig string

를 지정합니다 `KUBECONFIG` 로컬로 또는 Kubernetes 클러스터 간에 명령을 실행할 수 있는 경로입니다.

> 참고 또는 를 내보낼 수 있습니다 `KUBECONFIG` 특정 Kubernetes 클러스터 및 문제를 가리키는 변수 `tridentctl` 명령을 제공할 수 있습니다.

-n, --namespace string

Trident 배포의 네임스페이스입니다.

-o, --output string

출력 형식. json|YAML|name|wide|ps(기본값) 중 하나.

-s, --server string

Trident REST 인터페이스의 주소/포트입니다.

경고	Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 및 서비스하도록 구성할 수 있습니다.

명령 옵션 및 플래그

생성

명령을 사용하여 `create` Trident에 리소스를 추가할 수 있습니다.

```
tridentctl create [option]
```

옵션

`backend`: Trident에 백엔드를 추가합니다.

삭제

명령을 사용하여 `delete` Trident에서 하나 이상의 리소스를 제거할 수 있습니다.

```
tridentctl delete [option]
```

옵션

`backend`: Trident에서 하나 이상의 저장소 백엔드를 삭제합니다.
`snapshot`: Trident에서 하나 이상의 볼륨 스냅샷을 삭제합니다.
`storageclass`: Trident에서 하나 이상의 저장소 클래스를 삭제합니다.
`volume`: Trident에서 하나 이상의 저장소 볼륨을 삭제합니다.

가져오기

명령을 사용하여 `get` 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.

깃발

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

이미지

`images`플래그를 사용하여 Trident에 필요한 컨테이너 이미지 테이블을 인쇄합니다.

```
tridentctl images [flags]
```

깃발

-h, --help`이미지 도움말.
`-v, --k8s-version string: Kubernetes 클러스터의 시맨틱 버전입니다.

볼륨 가져오기

명령을 사용하여 import volume 기존 볼륨을 Trident로 가져옵니다.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

별칭

volume, v

깃발

-f, --filename string`YAML 또는 JSON PVC 파일로 이동합니다.
`-h, --help: 볼륨에 대한 도움말입니다.
--no-manage:PV/PVC만 생성 볼륨 라이프사이클 관리를 가정하지 마십시오.

설치합니다

`install`플래그를 사용하여 Trident를 설치합니다.

```
tridentctl install [flags]
```

깃발

--autosupport-image string: Autosupport Telemetry의 컨테이너 이미지(기본값 "netapp/trident autosupport:<현재 버전>").
--autosupport-proxy string : Autosupport Telemetry를 보내기 위한 프록시의 주소/포트입니다.
--enable-node-prep : 노드에 필요한 패키지를 설치해 보세요.
--generate-custom-yaml : 아무것도 설치하지 않고 YAML 파일을 생성합니다.
-h , --help : 설치에 대한 도움말.
--http-request-timeout : Trident 컨트롤러의 REST API에 대한 HTTP 요청 시간 초과를 재정의합니다 (기본값 1분 30초).
--image-registry string : 내부 이미지 레지스트리의 주소/포트.
--k8s-timeout duration : 모든 Kubernetes 작업에 대한 시간 초과(기본값 3분 0초).
--kubelet-dir string : kubelet의 내부 상태의 호스트 위치(기본값 "/var/lib/kubelet").
--log-format string : Trident 로깅 형식(텍스트, JSON)(기본값은 "텍스트")입니다.
--node-prep : Trident 지정된 데이터 저장 프로토콜을 사용하여 볼륨을 관리하도록 Kubernetes 클러스터의 노드를 준비할 수 있도록 합니다. 현재, **iscsi** 지원되는 유일한 값입니다**. OpenShift 4.19**부터 이 기능을 지원하는 최소 **Trident** 버전은 **25.06.1**입니다**.**
--pv string : Trident 에서 사용되는 레거시 PV의 이름으로, 이것이 존재하지 않도록 합니다(기본값은 "trident").
--pvc string : Trident 에서 사용하는 레거시 PVC의 이름으로, 이것이 존재하지 않도록 합니다(기본값은

"trident").
`--silence-autosupport` : NetApp 에 자동 지원 번들을 자동으로 보내지 않습니다(기본값 true).
`--silent` : 설치 중에 대부분의 출력을 비활성화합니다.
`--trident-image string` : 설치할 Trident 이미지입니다.
`--k8s-api-qps` : Kubernetes API 요청에 대한 초당 쿼리 수(QPS) 제한(기본값 100, 선택 사항).
`--use-custom-yaml` : 설치 디렉토리에 있는 기존 YAML 파일을 사용합니다.
`--use-ipv6` : Trident 통신에 IPv6를 사용합니다.

로그

`logs`플래그를 사용하여 `Trident`에서 로그를 인쇄합니다.

```
tridentctl logs [flags]
```

깃발

`-a, --archive`: 별도로 지정하지 않는 한 모든 로그를 사용하여 지원 아카이브를 생성합니다.
`-h, --help`:로그에 대한 도움말입니다.
`-l, --log string`:표시할 Trident 로그. Trident|auto|Trident-operator|all 중 하나(기본값 "auto").
`--node string`: 노드 Pod 로그를 수집할 Kubernetes 노드 이름입니다.
`-p, --previous`: 이전 컨테이너 인스턴스가 있는 경우 로그를 가져옵니다.
`--sidecars`: 사이드카 컨테이너에 대한 로그를 가져옵니다.

전송

명령을 사용하여 `send` Trident에서 리소스를 보냅니다.

```
tridentctl send [option]
```

옵션

`autosupport`AutoSupport 아카이브를 NetApp으로 전송합니다.

설치 제거

`uninstall`플래그를 사용하여 `Trident`를 제거합니다.

```
tridentctl uninstall [flags]
```

깃발

`-h, --help`: 제거 도움말입니다.
`--silent`: 제거 중 대부분의 출력을 비활성화합니다.

업데이트

명령을 사용하여 `update` Trident에서 리소스를 수정합니다.

```
tridentctl update [option]
```

옵션

backend: Trident에서 백엔드를 업데이트합니다.

백엔드 상태를 업데이트합니다

를 사용합니다 update backend state 백엔드 작업을 일시 중지하거나 재개하는 명령입니다.

```
tridentctl update backend state <backend-name> [flag]
```

고려해야 할 사항

- TridentBackendConfig(tbc)를 사용하여 백엔드를 생성한 경우 파일을 사용하여 백엔드를 업데이트할 수 backend.json 없습니다.
- 가 tbc에 설정된 경우 userState 명령을 사용하여 수정할 수 없습니다 tridentctl update backend state <backend-name> --user-state suspended/normal .
- tbc를 통해 설정한 후 via tridentctl을 userState 다시 설정하려면 userState tbc에서 필드를 제거해야 합니다. 이 작업은 명령을 사용하여 수행할 수 kubectl edit tbc 있습니다. 필드가 제거된 후 userState 명령을 사용하여 백엔드의 을 변경할 userState 수 있습니다 tridentctl update backend state.
- 를 사용하여 tridentctl update backend state 를 userState`변경합니다. 또는 파일을 사용하여 업데이트할 수도 `userState TridentBackendConfig backend.json 있습니다. 이렇게 하면 백엔드의 완전한 재초기화가 트리거되고 시간이 오래 걸릴 수 있습니다.

  깃발

  -h, --help: 백엔드 상태에 대한 도움말입니다.
  --user-state: 로 설정합니다 suspended 백엔드 작업을 일시 중지합니다. 를 로 설정합니다 normal 백엔드 작업을 재개합니다. 를 로 설정한 경우 suspended:

- AddVolume 그리고 Import Volume 일시 중지되었습니다.
- CloneVolume ResizeVolume, , PublishVolume, , , , UnPublishVolume CreateSnapshot GetSnapshot RestoreSnapshot, , DeleteSnapshot, , , , RemoveVolume GetVolumeExternal ReconcileNodeAccess 사용 가능 상태를 유지합니다.

백엔드 구성 파일 또는 의 필드를 사용하여 백엔드 상태를 업데이트할 수도 userState TridentBackendConfig `backend.json`있습니다. 자세한 내용은 및 을 "백엔드 관리 옵션" "kubeck을 사용하여 백엔드 관리 수행"참조하십시오.

- 예: *

JSON을 참조하십시오

파일을 사용하여 를 업데이트하려면 다음 단계를 userState backend.json 수행하십시오.

1. backend.json`값이 'uspended'로 설정된 필드를 포함하도록 파일을 `userState 편집합니다.
2. 백엔드를 업데이트하려면 다음을 사용하세요. tridentctl update backend 명령 및 업데이트 경로 backend.json 파일.

   예: tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

명령을 사용하여 tbc를 적용한 후 편집할 수 `kubectl edit <tbc-name> -n <namespace>` 있습니다. 다음 예에서는 옵션을 사용하여 백엔드 상태를 일시 중단하도록 업데이트합니다 `userState: suspended` .

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

버전

사용 `version` 플래그를 사용하여 의 버전을 인쇄합니다 `tridentctl` 및 실행 중인 Trident 서비스를 제공합니다.

```
tridentctl version [flags]
```

깃발

`--client`: 클라이언트 버전만(서버가 필요하지 않음).
`-h, --help`: 버전에 대한 도움말입니다.

플러그인 지원

Tridentctl은 kubectl과 유사한 플러그인을 지원합니다. Tridentctl은 플러그인 바이너리 파일 이름이 "tridentctl-<plugin>" 체계를 따르고 바이너리가 경로 환경 변수를 나열한 폴더에 있는 경우 플러그인을 감지합니다. 검색된 모든 플러그인은 tridentctl 도움말의 플러그인 섹션에 나열됩니다. 필요한 경우 환경 변수 TRIDENTCTL_PLUGIN_PATH에

플러그인 폴더를 지정하여 검색을 제한할 수도 있습니다(예: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). 변수가 사용되는 경우, tridenctl 은 지정된 폴더에서만 검색합니다.

Trident 스토리지 운영 모니터링

Trident는 Trident 성능을 모니터링하는 데 사용할 수 있는 Prometheus 메트릭 엔드포인트 세트를 제공합니다.

개요

Trident에서 제공하는 메트릭을 사용하면 다음을 수행할 수 있습니다.

- Trident의 상태 및 구성을 지속적으로 확인합니다. 성공적인 작업이 어떻게 이루어지는지, 예상대로 백엔드와 통신할 수 있는지 확인할 수 있습니다.
- 백엔드 사용 정보를 검토하고 백엔드에서 프로비저닝되는 볼륨 수와 사용된 공간 등을 파악합니다.
- 사용 가능한 백엔드에 프로비저닝된 볼륨 양의 매핑을 유지합니다.
- 성과 추적. Trident가 백엔드와 통신하고 작업을 수행하는 데 걸리는 시간을 살펴볼 수 있습니다.

참고	기본적으로 Trident의 메트릭은 대상 포트에 노출됩니다. `8001` 에서 `/metrics` 종점. 이러한 측정항목은 Trident 가 설치되면 기본적으로 활성화됩니다. HTTPS를 통해 포트에서 Trident 메트릭을 사용하도록 구성할 수 있습니다. `8444` 또한.

필요한 것

- Trident가 설치된 Kubernetes 클러스터
- 프로메테우스(Prometheus) 인스턴스. 이것은 일 수 있습니다 "컨테이너형 Prometheus 구축" 또는 Prometheus를 로 실행하도록 선택할 수 있습니다 "네이티브 애플리케이션".

1단계: Prometheus 목표를 정의합니다

Prometheus 대상을 정의하여 Trident 관리하는 백엔드, Trident가 생성하는 볼륨 등에 대한 메트릭과 정보를 수집해야 합니다. 보다"Prometheus Operator 문서" .

2단계: Prometheus ServiceMonitor를 만듭니다

Trident 메트릭을 소비하려면 트리덴트 CSI 서비스를 감시하고 메티우스 포트에서 수신 대기하는 프로메테우스 서비스 모니터를 만들어야 합니다. 샘플 ServiceMonitor의 모양은 다음과 같습니다.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
```

```
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

이 ServiceMonitor 정의는 다음에 의해 반환된 메트릭을 검색합니다. `trident-csi` 서비스를 특별히 찾습니다 `metrics` 서비스의 종료점. 결과적으로 Prometheus는 이제 Trident의 측정 항목을 이해하도록 구성되었습니다.

kubelet은 Trident에서 직접 사용할 수 있는 메트릭과 더불어 자체 메트릭 엔드포인트를 통해 많은 메트릭을 노출합니다. `kubelet_volume_*` Kubelet는 연결된 볼륨, Pod 및 처리하는 기타 내부 작업에 대한 정보를 제공할 수 있습니다. 을 "여기"참조하십시오.

HTTPS를 통해 Trident 메트릭 사용

HTTPS(포트 8444)를 통해 Trident 메트릭을 사용하려면 TLS 구성을 포함하도록 ServiceMonitor 정의를 수정해야 합니다. 또한 다음을 복사해야 합니다. `trident-csi` 의 비밀 `trident` Prometheus가 실행되는 네임스페이스에 대한 네임스페이스입니다. 다음 명령을 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 메트릭에 대한 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
```

Trident tridentctl, Helm 차트, Operator 등 모든 설치 방법에서 HTTPS 메트릭을 지원합니다.

- 당신이 사용하는 경우 `tridentctl install` 명령을 전달하면 `--https-metrics` HTTPS 메트릭을 활성화하는 플래그입니다.
- Helm 차트를 사용하는 경우 다음을 설정할 수 있습니다. `httpsMetrics` HTTPS 메트릭을 활성화하는 매개변수입니다.
- YAML 파일을 사용하는 경우 다음을 추가할 수 있습니다. `--https_metrics` 에 깃발을 `trident-main` 컨테이너에 `trident-deployment.yaml` 파일.

3단계: PromQL을 사용하여 Trident 메트릭 쿼리

PromQL은 시계열 또는 표 형식 데이터를 반환하는 식을 만드는 데 적합합니다.

다음은 사용할 수 있는 몇 가지 PromQL 쿼리입니다.

Trident 상태 정보를 가져옵니다

- **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 코드를 통한 **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 지속 시간**(ms)**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Trident 사용 정보를 가져옵니다

- 평균 볼륨 크기

```
trident_volume_allocated_bytes/trident_volume_count
```

- 각 백엔드에서 프로비저닝된 총 볼륨 공간

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

개별 볼륨 사용량을 가져옵니다

참고 | 이 기능은 kubelet 메트릭도 수집한 경우에만 사용할 수 있습니다.

- 각 볼륨에 사용된 공간의 비율

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Trident AutoSupport 원격 측정 기능에 대해 알아봅니다

기본적으로 Trident는 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp에 매일 보냅니다.

- Trident에서 Prometheus 메트릭 및 기본 백엔드 정보를 NetApp로 보내지 않도록 하려면 `--silence -autosupport` Trident 설치 중에 플래그를 전달합니다.
- 또한 Trident는 를 통해 컨테이너 로그를 NetApp 지원 팀에 온디맨드로 전송할 수 `tridentctl send autosupport` 있습니다. 로그를 업로드하려면 Trident를 트리거해야 합니다. 로그를 제출하기 전에 NetApp를 수락해야 "개인 정보 보호 정책"합니다.
- 별도로 지정하지 않는 한 Trident는 지난 24시간의 로그를 가져옵니다.
- 플래그를 사용하여 로그 보존 기간을 지정할 수 `--since` 있습니다. 예를 들면 다음과 `tridentctl send autosupport --since=1h`같습니다. 이 정보는 `Trident`와 함께 설치된 컨테이너를 통해 수집되고 전송됩니다 `trident-autosupport`. 컨테이너 이미지는 에서 얻을 수 "Trident AutoSupport를 누릅니다" 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. 개인 레지스트리를 사용하여 컨테이너 이미지를 저장하는 경우 '--image-registry' 플래그를 사용할 수 있습니다.
- 또한 설치 YAML 파일을 생성하여 프록시 URL을 구성할 수도 있습니다. 이는 트라이덴트ctl install --generate -custom-YAML을 이용해 YAML 파일을 생성하고 트리덴트 배포(trident-deployment)의 트리덴트 자동 지원 컨테이너에 대한 '--proxy-url' 주장을 추가하는 방식으로 가능하다.

Trident 메트릭을 비활성화합니다

- 메트릭을 보고하지 않으려면 ('--generate-custom-YAML' 플래그를 사용하여) 사용자 지정 YAML을 생성하고 이를 편집하여 삼중류-main' 컨테이너에 대해 호출되는 '--metrics' 플래그를 제거해야 합니다.

Trident를 제거합니다

Trident를 설치하는 데 사용한 Trident를 제거하는 것과 동일한 방법을 사용해야 합니다.

이 작업에 대해

- 업그레이드, 종속성 문제 또는 실패하거나 불완전한 업그레이드 후에 발견된 버그에 대한 수정이 필요한 경우 Trident를 제거하고 해당 지침에 따라 이전 버전을 다시 설치해야 "버전"합니다. 이전 버전으로 _download_하는 유일한 권장 방법입니다.
- 간편한 업그레이드 및 재설치를 위해 Trident를 제거해도 Trident에서 생성한 CRD 또는 관련 개체는 제거되지 않습니다. Trident 및 모든 해당 데이터를 완전히 제거해야 하는 경우 을 참조하십시오"Trident 및 CRD를 완전히 제거합니다".

시작하기 전에

Kubernetes 클러스터를 사용 중단하는 경우, 를 제거하기 전에 Trident에서 만든 볼륨을 사용하는 모든 애플리케이션을 삭제해야 합니다. 따라서 PVC가 삭제되기 전에 Kubernetes 노드에 게시되지 않습니다.

원래 설치 방법을 확인합니다

Trident를 설치할 때 사용한 것과 동일한 방법을 사용해야 합니다. 제거하기 전에 Trident를 처음 설치할 때 사용한 버전을 확인하십시오.

1. 사용 `kubectl get pods -n trident` 를 눌러 포드를 검사합니다.
   - 운영자 포드가 없으면 를 사용하여 Trident를 설치한 `tridentctl` 것입니다.
   - 운영자 포드가 있는 경우, Trident는 Trident 운영자를 사용하여 수동으로 또는 Helm을 사용하여 설치되었습니다.
2. 운영자 포드가 있는 경우 를 `kubectl describe tproc trident` 사용하여 Trident가 Helm을 사용하여 설치되었는지 확인합니다.
   - Helm 레이블이 있는 경우 Helm을 사용하여 Trident를 설치한 것입니다.
   - Helm 레이블이 없는 경우 Trident 연산자를 사용하여 Trident를 수동으로 설치했습니다.

Trident 운영자 설치를 제거합니다

수동 또는 Helm을 사용하여 트라이덴트 작업자 설치를 제거할 수 있습니다.

수동 설치를 제거합니다

운영자를 사용하여 Trident를 설치한 경우 다음 중 하나를 수행하여 제거할 수 있습니다.

1. 편집 **`TridentOrchestrator` CR** 및 제거 플래그 설정:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"uninstall":true}}'
```

   를 누릅니다 `uninstall` 플래그가 로 설정되어 있습니다 `true`Trident 운영자가 Trident를 제거하지만 Trident 자체 자체를 제거하지는 않습니다. Trident를 다시 설치하려면 해당 Trident를 정리하고 새 AgentOrchestrator를 생성해야 합니다.

2. 삭제 **`TridentOrchestrator`**: Trident를 배포하는 데 사용된 CR을 제거하면 `TridentOrchestrator` 운영자에게 Trident를 제거하도록 지시합니다. 운영자는 의 제거를 `TridentOrchestrator` 처리하고 Trident 배포 및 데몬 세트를 제거하며 설치의 일부로 생성한 Trident 포드를 삭제합니다.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Helm 설치를 제거합니다

Helm을 사용하여 Trident를 설치한 경우 을 사용하여 제거할 수 `helm uninstall` 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
```

를 제거합니다 **tridentctl** 설치

`uninstall`CRD 및 관련 객체를 제외하고 Trident와 연결된 모든 리소스를 제거하려면 의 명령을 `tridentctl` 사용합니다.

```
./tridentctl uninstall -n <namespace>
```

Docker를 위한 Trident

배포를 위한 사전 요구 사항

Trident를 배포하기 전에 호스트에 필요한 프로토콜 사전 요구 사항을 설치하고 구성해야 합니다.

요구 사항을 확인합니다

- 배포가 모든 를 충족하는지 확인합니다 "요구 사항".
- 지원되는 버전의 Docker가 설치되어 있는지 확인합니다. Docker 버전이 오래되었으면 "설치 또는 업데이트합니다".

```
docker --version
```

- 프로토콜 사전 요구 사항이 호스트에 설치 및 구성되어 있는지 확인합니다.

NFS 툴

운영 체제의 명령을 사용하여 NFS 툴을 설치합니다.

RHEL 8+

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

우분투

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

경고	볼륨에 연결할 때 오류가 발생하지 않도록 NFS 툴을 설치한 후 작업자 노드를 재부팅합니다.

iSCSI 툴

운영 체제의 명령을 사용하여 iSCSI 도구를 설치합니다.

RHEL 8+

1. 다음 시스템 패키지를 설치합니다.

```
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 확인합니다.

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

3. 스캔을 수동으로 설정합니다.

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

4. 다중 경로 설정:

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

참고 | etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다.

5. iscsid와 multipathd가 실행 중인지 확인합니다.

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

6. "iSCSI" 활성화 및 시작:

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

우분투

1. 다음 시스템 패키지를 설치합니다.

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

2. open-iscsi 버전이 2.0.874-5ubuntu2.10 이상(bionic) 또는 2.0.874-7.1uubuttu6.1 이상(focal)인지 확인합니다.

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

3. 스캔을 수동으로 설정합니다.

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

4. 다중 경로 설정:

```
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
```

참고 etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다.

5. 'open-iscsi'와 'multirpath-tools'가 활성화되어 실행되고 있는지 확인합니다.

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

NVMe 툴

운영 체제의 명령을 사용하여 NVMe 툴을 설치합니다.

참고

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

RHEL 9 를 참조하십시오

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

우분투

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

FC 도구

운영 체제의 명령을 사용하여 FC 툴을 설치합니다.

- RHEL/Red Hat Enterprise Linux CoreOS(RHCOS)를 실행하는 작업자 노드를 FC PVS와 함께 사용하는 경우 StorageClass에서 mount옵션을 지정하여 `discard` 인라인 공간 재확보를 수행합니다. 을 "Red Hat 설명서 "참조하십시오.

RHEL 8+

1. 다음 시스템 패키지를 설치합니다.

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

2. 다중 경로 설정:

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

참고 | etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다.

3. 가 `multipathd` 실행 중인지 확인합니다.

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

우분투

1. 다음 시스템 패키지를 설치합니다.

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

2. 다중 경로 설정:

```
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
```

참고 | etc/multipath.conf에 debrofs 아래에 find_multiprohs no가 포함되어 있는지 확인합니다.

3. 가 활성화되어 있고 실행 중인지 `multipath-tools` 확인합니다.

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

Trident를 구축합니다

Trident for Docker는 NetApp 스토리지 플랫폼을 위한 Docker 에코시스템과 직접 통합됩니다. 또한 향후 추가 플랫폼을 추가할 수 있도록 스토리지 플랫폼에서 Docker 호스트에 이르는 스토리지 리소스의 프로비저닝 및 관리를 지원합니다.

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

를 "배포를 위한 사전 요구 사항"참조하십시오. 사전 요구 사항이 충족되었는지 확인한 후 Trident를 배포할 준비가 된 것입니다.

Docker 관리형 플러그인 방법(버전 1.13/17.03 이상)

참고	시작하기 전에 기존 데몬 메서드에서 Trident pre Docker 1.13/17.03을 사용한 경우 관리되는 플러그인 방법을 사용하기 전에 Trident 프로세스를 중지하고 Docker 데몬을 다시 시작해야 합니다.

1. 실행 중인 모든 인스턴스 중지:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Docker를 다시 시작합니다.

```
systemctl restart docker
```

3. Docker Engine 17.03(새로운 1.13) 이상이 설치되어 있는지 확인합니다.

```
docker --version
```

버전이 최신 버전이 아니면 "설치를 설치하거나 업데이트합니다".

단계

1. 구성 파일을 생성하고 다음과 같이 옵션을 지정합니다.
   - config 기본 파일 이름은 config.json 이지만 원하는 이름은 파일 이름으로 config 옵션을 지정하여 사용할 수

있습니다. 구성 파일은 호스트 시스템의 '/etc/netapdvp' 디렉토리에 있어야 합니다.

- 로그 수준: 로깅 수준(debug, info, warn, error, fatal)을 지정합니다. 기본값은 '정보'입니다.
- debug: 디버그 로깅을 사용할 수 있는지 여부를 지정합니다. 기본값은 false 입니다. TRUE인 경우 로그 수준을 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. 관리되는 플러그인 시스템을 사용하여 Trident를 시작합니다. `<version>`사용 중인 플러그인 버전(xxx.xx.x)으로 바꿉니다.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Trident를 사용하여 구성된 시스템에서 스토리지를 소비합니다.

a. "firstVolume"이라는 이름의 볼륨을 생성합니다.

```
docker volume create -d netapp --name firstVolume
```

b. 컨테이너가 시작될 때 기본 볼륨을 생성합니다.

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. "firstVolume" 볼륨을 제거합니다.

```
docker volume rm firstVolume
```

기존 방법(버전 **1.12** 이하)

시작하기 전에

1. Docker 버전 1.10 이상이 설치되어 있는지 확인합니다.

```
docker --version
```

버전이 최신 버전이 아니면 설치를 업데이트하십시오.

```
curl -fsSL https://get.docker.com/ | sh
```

또는, "배포 지침을 따릅니다".

2. 시스템에 NFS 및/또는 iSCSI가 구성되어 있는지 확인합니다.

단계

1. NetApp Docker Volume Plugin 설치 및 구성:

a. 응용 프로그램 다운로드 및 압축 풀기:

```
----trident-installer-26.10.1.tar.gz
wget https://github.com/NetApp/trident/releases/download/10.0/
tar zxf trident-installer-26.10.1.tar.gz
```

```
.. Move to a location in the bin path:
+
[source,console]
```

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

```
.. Create a location for the configuration file:
+
[source,console]
```

sudo mkdir -p /etc/netappdvp

```
.. Create the configuration file:
+
[source,console]
```

cat << EOF > /etc/netappdvp/ontap-nas.json

```
+
[source,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

```
. After placing the binary and creating the configuration file, start the
Trident daemon using the desired configuration file.
+
[source,console]
```

sudo trident --config=/etc/netappdvp/ontap-nas.json

```
+
NOTE: Unless specified, the default name for the volume driver is
"netapp".
+
After the daemon is started, you can create and manage volumes by using
the Docker CLI interface.

. Create a volume:
+
[source,console]
```

docker volume create -d netapp --name trident_1

```
. Provision a Docker volume when starting a container:
+
[source,console]
```

docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol alpine ash

```
. Remove a Docker volume:
+
```

```
[source,console]
```

docker volume rm trident_1

```
+
[source,console]
```

docker volume rm trident_2

```
== Start Trident at system startup

A sample unit file for systemd based systems can be found at
`contrib/trident.service.example` in the Git repo. To use the file with
RHEL, do the following:

. Copy the file to the correct location.
+
You should use unique names for the unit files if you have more than one
instance running.
+
[source,console]
```

cp contrib/trident.service.example /usr/lib/systemd/system/trident.service

```
. Edit the file, change the description (line 2) to match the driver name
and the configuration file path (line 9) to reflect your environment.
. Reload systemd for it to ingest changes:
+
[source,console]
```

systemctl daemon-reload

```
. Enable the service.
+
This name varies depending on what you named the file in the
`/usr/lib/systemd/system` directory.
+
[source,console]
```

systemctl enable trident

```
. Start the service.
```

```
+
[source,console]
```

systemctl start trident

```
. View the status.
+
[source,console]
```

systemctl status trident

```
NOTE: Any time you modify the unit file, run the `systemctl daemon-reload`
command for it to be aware of the changes.
```

Trident 업그레이드 또는 제거

Trident for Docker를 사용 중인 볼륨에 영향을 주지 않고 안전하게 업그레이드할 수 있습니다. 업그레이드 프로세스 중에는 플러그인에 지시된 명령이 성공하지 못하고 플러그인이 다시 실행될 때까지 응용 프로그램에서 볼륨을 마운트할 수 없는 짧은 기간이 `docker volume` 있습니다. 대부분의 경우 몇 초 이내에 완료됩니다.

업그레이드

Trident for Docker를 업그레이드하려면 아래 단계를 수행하십시오.

단계

1. 기존 볼륨 나열:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. 플러그인 비활성화:

```
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
```

참고 | nDVP가 Trident 18.01 릴리즈로 대체됩니다. 이미지에서 이미지로 `netapp/trident-plugin` 직접 업그레이드해야 `netapp/ndvp-plugin` 합니다.

4. 플러그인 활성화:

```
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
```

중요함 | 이전 버전의 Trident(20.10 이전)에서 Trident 20.10 이상으로 업그레이드하는 경우 오류가 발생할 수 있습니다. 자세한 내용은 을 "알려진 문제"참조하십시오. 오류가 발생하면 먼저 플러그인을 비활성화한 다음 플러그인을 제거한 다음 추가 구성 매개 변수를 전달하여 필요한 Trident 버전을 설치해야 합니다. `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

설치 제거

Trident for Docker를 제거하려면 아래 단계를 수행하십시오.

단계

1. 플러그인이 생성한 모든 볼륨을 제거합니다.
2. 플러그인 비활성화:

```
docker plugin disable netapp:latest
```

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. 플러그인 제거:

```
docker plugin rm netapp:latest
```

볼륨 작업

필요한 경우 Trident 드라이버 이름을 지정한 표준 명령을 사용하여 볼륨을 쉽게 생성, 클론 복제 및 제거할 수 `docker volume` 있습니다.

볼륨을 생성합니다

- 기본 이름을 사용하여 드라이버로 볼륨을 생성합니다.

```
docker volume create -d netapp --name firstVolume
```

- 특정 Trident 인스턴스를 사용하여 볼륨 생성:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

참고 지정하지 않은 경우 "옵션", 드라이버 기본값이 사용됩니다.

- 기본 볼륨 크기를 재정의합니다. 드라이버를 사용하여 20GiB 볼륨을 생성하는 다음 예를 참조하세요.

```
docker volume create -d netapp --name my_vol --opt size=20G
```

팁 볼륨 크기는 옵션 단위(예: 10G, 20GB, 3TiB)가 포함된 정수 값이 포함된 문자열로 표시됩니다. 단위를 지정하지 않으면 기본값은 G입니다 크기 단위는 2(B, KiB, MiB, GiB, TiB) 또는 10(B, KB, MB, GB, TB)의 거듭제곱으로 표현될 수 있습니다. 단축 단위는 2의 거듭제곱을 사용합니다(G=GiB, T=TiB,…).

볼륨을 제거합니다

- 다른 Docker 볼륨과 마찬가지로 볼륨을 제거합니다.

```
docker volume rm firstVolume
```

중요함 | '솔드파이어-SAN' 드라이버를 사용할 때 위의 예에서는 볼륨을 삭제하고 삭제합니다.

Trident for Docker를 업그레이드하려면 아래 단계를 수행하십시오.

볼륨의 클론을 생성합니다

사용 시 `ontap-nas` , `ontap-san` , 그리고 `solidfire-san` 저장 드라이버인 Trident 볼륨을 복제할 수 있습니다. 사용 시 `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버, 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새로운 스냅샷이 생성됩니다.

- 볼륨을 검사하여 스냅샷을 열거합니다.

```
docker volume inspect <volume_name>
```

- 기존 볼륨에서 새 볼륨을 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 스냅샷은 생성하지 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 액세스합니다

파티션이 없고 파일 시스템이 Trident에서 지원되는 경우(예: Trident를 통해 형식 지정된 `/dev/sdc1` 블록 장치에 액세스할 수 없음) Trident * only * 를 사용하여 컨테이너를 통해 외부에서 생성된 블록 장치(또는 해당 클론) `ext4`에 액세스할 수 있습니다.

드라이버별 볼륨 옵션

각 스토리지 드라이버에는 다양한 옵션이 있으며, 볼륨 생성 시 이를 지정하여 결과를 사용자 지정할 수 있습니다. 구성된 스토리지 시스템에 적용되는 옵션은 아래를 참조하십시오.

볼륨 생성 작업 중에 이러한 옵션을 사용하는 것은 간단합니다. CLI 동작 중에 '-o' 연산자를 사용하여 옵션과 값을 제공한다. 이러한 값은 JSON 구성 파일의 모든 등가 값을 재정의합니다.

ONTAP 볼륨 옵션

NFS, iSCSI 및 FC 모두의 볼륨 생성 옵션에는 다음이 포함됩니다.

옵션을 선택합니다	설명
'크기'입니다	볼륨의 크기는 기본적으로 1GiB로 설정됩니다.
'예비공간'	볼륨을 씬 또는 일반 프로비저닝합니다. 기본값은 Thin 입니다. 유효한 값은 '없음'(씬 프로비저닝) 및 '볼륨'(일반 프로비저닝)입니다.

옵션을 선택합니다	설명
냅샷정책	그러면 스냅샷 정책이 원하는 값으로 설정됩니다. 기본값은 `none` 볼륨에 대해 스냅샷이 자동으로 생성되지 않는다는 의미입니다. 스토리지 관리자가 수정하지 않는 한 모든 ONTAP 시스템에 "default"라는 정책이 존재하며, 이 정책은 6시간, 2일, 2주 간격으로 스냅샷을 생성하고 유지합니다. 스냅숏에 보존된 데이터는 볼륨의 임의의 디렉터리에 있는 디렉터리로 이동하여 복구할 수 `.snapshot` 있습니다.
안산예비역	이렇게 하면 스냅숏 예비 공간이 원하는 비율로 설정됩니다. 기본값은 값이 없습니다. 즉, snapshotPolicy를 선택한 경우 ONTAP가 snapshotReserve(일반적으로 5%)를 선택하거나 snapshotPolicy가 none인 경우 0%를 선택합니다. 모든 ONTAP 백엔드에 대한 구성 파일에서 기본 snapshotReserve 값을 설정할 수 있으며 ONTAP-NAS-이코노미 를 제외한 모든 ONTAP 백엔드에 대한 볼륨 생성 옵션으로 사용할 수 있습니다.
'plitOnClone'을 선택합니다	볼륨을 클론 생성할 때 ONTAP가 상위 클론에서 즉시 클론을 분할합니다. 기본값은 입니다 `false`. 볼륨을 클론 복제하는 사용 사례에는 스토리지 효율성을 높일 기회가 없을 것 같기 때문에 생성 즉시 클론을 상위 볼륨에서 분리하는 것이 가장 좋습니다. 예를 들어, 빈 데이터베이스를 복제하면 시간이 절약되지만 스토리지가 거의 절약되지 않을 수 있으므로 클론을 즉시 분할하는 것이 가장 좋습니다.
암호화	새 볼륨에 NVE(NetApp Volume Encryption)를 사용하도록 설정하고 기본값은 'false'입니다. 이 옵션을 사용하려면 NVE 라이센스가 클러스터에서 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 사용됩니다. 자세한 내용은 다음을 "Trident가 NVE 및 NAE와 작동하는 방법"참조하십시오.
'계층화 정책'	볼륨에 사용할 계층화 정책을 설정합니다. 비활성(콜드) 상태일 때 데이터를 클라우드 계층으로 이동할지 결정합니다.

다음 추가 옵션은 NFS * 에만 적용됩니다 *.

옵션을 선택합니다	설명
유니크권한	볼륨 자체에 대한 권한 집합을 제어합니다. 기본적으로 사용 권한은 '---rwxr-XR-x' 또는 숫자 표기 0755로 설정되며 root는 소유자가 됩니다. 텍스트 또는 숫자 형식이 작동합니다.

옵션을 선택합니다	설명
나프산디렉토리	를 로 설정합니다 `true` 가 을(를) 만듭니다 `.snapshot` 볼륨을 액세스하는 클라이언트가 볼 수 있는 디렉토리입니다. 기본값은 입니다 `false`이는 의 가시성을 의미합니다 `.snapshot` 디렉토리는 기본적으로 사용되지 않습니다. 공식 MySQL 이미지와 같은 일부 이미지는 에서 예상대로 작동하지 않습니다 `.snapshot` 디렉토리가 표시됩니다.
엑포트정책	볼륨에 사용할 엑스포트 정책을 설정합니다. 기본값은 '기본값'입니다.
'생태성 스타일'을 참조하십시오	볼륨에 액세스하는 데 사용할 보안 스타일을 설정합니다. 기본값은 UNIX입니다. 유효한 값은 UNIX와 Mixed 입니다.

다음 추가 옵션은 iSCSI * 에만 적용됩니다 *.

옵션을 선택합니다	설명
파일 시스템 유형	iSCSI 볼륨을 포맷하는 데 사용되는 파일 시스템을 설정합니다. 기본값은 ext4 입니다. 유효한 값은 ext3, ext4, xfs입니다.
'팩시배부	를 로 설정합니다 `false` 에서 LUN의 공간 할당 기능을 해제합니다. 기본값은 입니다 `true`즉, ONTAP는 볼륨에 공간이 부족하고 볼륨의 LUN이 쓰기를 허용할 수 없을 때 호스트에 알립니다. 또한 이 옵션을 사용하면 호스트가 데이터를 삭제할 때 ONTAP에서 자동으로 공간을 재확보할 수 있습니다.

예

아래 예를 참조하십시오.

- 10GiB 볼륨을 생성합니다.

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- 스냅샷으로 100GiB 볼륨을 만듭니다.

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- setuid 비트가 설정된 볼륨을 생성합니다.

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

최소 볼륨 크기는 20MiB입니다.

스냅숏 예비 공간이 지정되지 않고 스냅숏 정책이 인 경우 `none` Trident는 0%의 스냅숏 예비 공간을 사용합니다.

- 스냅숏 정책이 없고 스냅숏 예비 공간이 없는 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- 스냅샷 정책이 없는 볼륨 및 10%의 사용자 지정 스냅숏 예비 공간을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- 스냅샷 정책 및 10%의 사용자 지정 스냅숏 예비 공간이 있는 볼륨을 생성합니다.

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- 스냅샷 정책을 사용하여 볼륨을 생성하고 ONTAP의 기본 스냅숏 예비 공간을 수락합니다(일반적으로 5%).

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Element 소프트웨어 볼륨 옵션

Element 소프트웨어 옵션은 볼륨과 연관된 서비스 품질(QoS) 정책의 크기 및 크기를 표시합니다. 볼륨을 생성할 때 해당 볼륨과 연관된 QoS 정책은 '-o type=service_level' 명명법을 사용하여 지정합니다.

Element 드라이버로 QoS 서비스 수준을 정의하는 첫 번째 단계는 하나 이상의 유형을 생성하고 구성 파일의 이름과 연결된 최소, 최대 및 버스트 IOPS를 지정하는 것입니다.

기타 Element 소프트웨어 볼륨 생성 옵션에는 다음이 포함됩니다.

옵션을 선택합니다	설명
'크기'입니다	볼륨 크기는 기본적으로 1GiB 또는 구성 항목 … "defaults": {"size": "5G"}입니다.
블록사이즈	512 또는 4096 중 하나를 사용합니다. 기본값은 512 또는 구성 항목 DefaultBlockSize 입니다.

예

QoS 정의가 포함된 다음 샘플 구성 파일을 참조하십시오.

```
{
  "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 구성에서는 Bronze, Silver, Gold의 세 가지 정책 정의가 있습니다. 이러한 이름은 임의로 지정됩니다.

- 10GiB Gold 볼륨을 생성합니다.

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- 100GiB Bronze 볼륨을 생성합니다.

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

로그를 수집합니다

문제 해결에 도움이 되는 로그를 수집할 수 있습니다. 로그를 수집하는 방법은 Docker 플러그인을 실행하는 방법에 따라 다릅니다.

문제 해결을 위해 로그를 수집합니다

단계

1. 권장되는 관리되는 플러그인 방법(예: 명령 사용)을 사용하여 Trident를 실행하는 경우 `docker plugin` 다음과 같이 봅니다.

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

표준 로깅 수준에서는 대부분의 문제를 진단할 수 있어야 합니다. 이 방법으로는 충분하지 않으면 디버그 로깅을 활성화할 수 있습니다.

2. 디버그 로깅을 사용하려면 디버그 로깅을 사용하도록 설정한 플러그인을 설치합니다.

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

또는 플러그인이 이미 설치된 경우 디버그 로깅을 활성화합니다.

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. 호스트에서 바이너리 자체를 실행하는 경우 호스트의 에서 로그를 사용할 수 있습니다 `/var/log/netappdvp` 디렉토리. 디버그 로깅을 활성화하려면 를 지정합니다 `-debug` 플러그인을 실행할 때

일반적인 문제 해결 팁

- 새 사용자가 실행하는 가장 일반적인 문제는 플러그 인을 초기화할 수 없도록 잘못 구성된 것입니다. 이 경우 플러그인을 설치하거나 활성화하려고 할 때 다음과 같은 메시지가 표시될 수 있습니다.

  "데몬으로부터 오류 응답: UNIX/run/docker/plugins/<id>/NetApp.sock:connect:해당 파일 또는 디렉토리 없음

  즉, 플러그인을 시작하지 못했습니다. 다행히 플러그인은 사용자가 겪을 수 있는 대부분의 문제를 진단하는 데 도움이 되는 포괄적인 로깅 기능을 갖추고 있습니다.

- PV를 컨테이너에 마운트하는 데 문제가 있는 경우 rpcbind가 설치되어 실행되고 있는지 확인합니다. 호스트 OS에 필요한 패키지 관리자를 사용하고 rpcbind가 실행 중인지 확인합니다. 'stemctl status rpcbind' 또는 이와 동등한 기능을 실행하여 rpcbind 서비스의 상태를 확인할 수 있습니다.

여러 **Trident** 인스턴스를 관리합니다

여러 스토리지 구성을 동시에 사용할 수 있도록 하려는 경우 Trident의 여러 인스턴스가 필요합니다. 여러 인스턴스의 핵심은 컨테이너형 플러그인과 함께 '- alias' 옵션을 사용하거나 호스트에서 Trident를 인스턴스화할 때 '- volume-driver' 옵션을 사용하여 다른 이름을 지정하는 것입니다.

Docker 관리 플러그인 단계(버전 **1.13/17.03** 이상)

1. 별칭 및 구성 파일을 지정하는 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 지정하는 볼륨을 생성합니다.

   예를 들어 금괴 볼륨의 경우:

```
docker volume create -d gold --name ntapGold
```

   예를 들어, 은 볼륨의 경우:

```
docker volume create -d silver --name ntapSilver
```

기존(버전 **1.12** 이하) 단계

1. 사용자 지정 드라이버 ID를 사용하여 NFS 구성으로 플러그인을 시작합니다.

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. 사용자 지정 드라이버 ID를 사용하여 iSCSI 구성으로 플러그인을 시작합니다.

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. 각 드라이버 인스턴스에 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 참조하십시오.

글로벌 구성 옵션

이러한 구성 옵션은 사용 중인 스토리지 플랫폼에 관계없이 모든 Trident 구성에 적용됩니다.

옵션을 선택합니다	설명	예
'내전'	구성 파일 버전 번호입니다	`1`
'torageDriverName'입니다	스토리지 드라이버 이름입니다	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
'토르agePrefix'	볼륨 이름에 대한 선택적 접두사입니다. 기본값: `netappdvp_`.	`staging_`
LimitVolumeSize	볼륨 크기에 대한 선택적 제한. 기본값:""(적용 안 됨)	`10g`

팁

Element 백엔드에 (기본값 포함)을 사용하지 `storagePrefix` 마십시오. 기본적으로 `solidfire-san` 드라이버는 이 설정을 무시하고 접두사를 사용하지 않습니다. NetApp은 Docker 볼륨 매핑에 특정 tenantID를 사용하거나, 이름과 같이 사용되었을 수 있는 경우 Docker 버전, 드라이버 정보 및 Docker의 원시 이름으로 채워진 특성 데이터를 사용할 것을 권장합니다.

생성하는 모든 볼륨에서 기본 옵션을 지정하지 않아도 됩니다. '크기' 옵션은 모든 컨트롤러 유형에 사용할 수 있습니다. 기본 볼륨 크기를 설정하는 방법은 ONTAP 구성 섹션을 참조하십시오.

옵션을 선택합니다	설명	예
'크기'입니다	새 볼륨의 선택적 기본 크기입니다. 기본값: `1G`	`10G`

ONTAP 구성

위의 글로벌 구성 값 외에도 ONTAP를 사용할 경우 다음과 같은 최상위 옵션을 사용할 수 있습니다.

옵션을 선택합니다	설명	예
마나멘타LIF	ONTAP 관리 LIF의 IP 주소입니다. FQDN(정규화된 도메인 이름)을 지정할 수 있습니다.	`10.0.0.1`
다타LIF	프로토콜 LIF의 IP 주소입니다. **ONTAP nas drivers**: NetApp는 지정할 것을 `dataLIF` 권장합니다. 제공되지 않는 경우 Trident는 SVM에서 데이터 LIF를 가져옵니다. NFS 마운트 작업에 사용할 FQDN(정규화된 도메인 이름)을 지정할 수 있습니다. 이렇게 하면 라운드 로빈 DNS를 생성하여 여러 데이터 LIF의 로드 밸런싱을 수행할 수 있습니다. **ONTAP SAN** 드라이버: iSCSI 또는 FC에 대해 지정하지 마십시오. Trident는 을 사용하여 "ONTAP 선택적 LUN 맵"다중 경로 세션을 설정하는 데 필요한 iSCSI 또는 FC LIF를 검색합니다. 이 명시적으로 정의된 경우 경고가 `dataLIF` 생성됩니다.	`10.0.0.2`
'VM'입니다	사용할 스토리지 가상 머신(관리 LIF가 클러스터 LIF인 경우 필요)	`svm_nfs`
'사용자 이름'	스토리지 디바이스에 접속할 사용자 이름입니다	`vsadmin`
"암호"	스토리지 디바이스에 연결하는 암호입니다	`secret`

옵션을 선택합니다	설명	예
골재	프로비저닝을 위한 애그리게이트(선택 사항, SVM에 셋팅해야 하는 경우) 드라이버의 경우 `ontap-nas-flexgroup` 이 옵션은 무시됩니다. SVM에 할당된 모든 애그리게이트는 FlexGroup 볼륨을 프로비저닝하는 데 사용됩니다.	`aggr1`
제한선태사용법	선택 사항으로, 사용량이 이 백분율보다 높을 경우 용량 할당에 실패합니다	`75%`
nfsMountOptions를 선택합니다	NFS 마운트 옵션을 세부적으로 제어하고 기본값은 "-o nfsvers=3"입니다. 및 **`ontap-nas-economy`** 드라이버에서만 사용 가능 **`ontap-nas`**. "NFS 호스트 구성 정보는 여기 를 참조하십시오"..	`-o nfsvers=4`
“인명이름”입니다	Trident는 노드당 를 `netappdvp` 생성하고 `igroups` 관리합니다. 이 값은 변경하거나 생략할 수 없습니다. • 에서만 사용할 수 있습니다 `ontap-san` 드라이버 *.	`netappdvp`
LimitVolumeSize	최대 요청 가능 볼륨 크기입니다.	`300g`
"케트리스퍼플랙스볼륨"	FlexVol당 최대 qtree는 범위 [50, 300]에 있어야 하며 기본값은 200입니다. 의 경우 * `ontap-nas-economy` 드라이버, 이 옵션을 사용하면 FlexVol*당 최대 qtree 수를 사용자 지정할 수 있습니다.	`300`
`sanType`	* `ontap-san`드라이버에서만 지원됩니다.* `iSCSI, `nvme` NVMe/TCP 또는 `fcp` FC(SCSI over Fibre Channel)를 선택할 때 `iscsi` 사용합니다.	`iscsi` 비어 있는 경우
`limitVolumePoolSize`	* `ontap-san-economy`및 `ontap-san-economy` 드라이버에서만 지원됩니다.* ONTAP ONTAP-NAS-Economy 및 ONTAP-SAN-Economy 드라이버의 FlexVol 크기를 제한합니다.	`300g`

생성하는 모든 볼륨에 기본 옵션을 지정하지 않아도 됩니다.

옵션을 선택합니다	설명	예
'예비공간'	공간 예약 모드; none (씬 프로비저닝됨) 또는 을 클릭합니다 volume (두꺼운)	"없음"
냅샷정책	사용할 스냅샷 정책이며 기본값은 입니다 none	"없음"
안산예비역	스냅샷 예약 비율입니다. 기본값은 ""이며 ONTAP 기본값을 적용합니다	10
'plitOnClone'을 선택합니다	생성 시 상위 클론에서 클론을 분할합니다. 기본값은 입니다 false	거짓입니다
암호화	새 볼륨에 NVE(NetApp Volume Encryption)를 지원하며 기본값은 'false'입니다. 이 옵션을 사용하려면 NVE 라이센스가 클러스터에서 활성화되어 있어야 합니다. 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 사용됩니다. 자세한 내용은 다음을 "Trident가 NVE 및 NAE와 작동하는 방법"참조하십시오.	참
유니크권한	프로비저닝된 NFS 볼륨에 대한 NAS 옵션, 기본값은 로 설정됩니다 777	777
나프산디렉토리	디렉토리에 액세스하기 위한 NAS .snapshot 옵션입니다.	`.snapshot` 디렉터리에 대한 액세스를 활성화 또는 비활성화합니다. 유효값: true, false. 기본 동작은 ONTAP 및 Trident 버전에 따라 다를 수 있습니다. 일관된 동작을 보장하려면 명시적으로 설정하십시오.
엑포트정책	NFS 엑스포트 정책에서 사용할 NAS 옵션이며 기본값은 입니다 default	default
'생태성 스타일'을 참조하십시오	프로비저닝된 NFS 볼륨에 액세스하기 위한 NAS 옵션입니다. NFS를 지원합니다 mixed 및 unix 보안 스타일. 기본값은 입니다 unix.	unix
파일 시스템 유형	SAN 옵션 - 파일 시스템 유형을 선택합니다. 기본값은 입니다 ext4	xfs
'계층화 정책'	사용할 계층화 정책입니다. 기본값은 입니다. none	"없음"
skipRecovery Queue	볼륨을 삭제하는 동안 저장소의 복구 대기열을 우회하고 볼륨을 즉시 삭제합니다.	``

축척 옵션

`ontap-nas` `ontap-san`및 드라이버는 각 Docker 볼륨에 대한 ONTAP FlexVol를 생성합니다. ONTAP는 클러스터 노드당 최대 1000개의 FlexVols 지원, 클러스터 최대 12,000개의 FlexVol 볼륨 요구 사항이 해당 제한 내에 포함되는 경우 `ontap-nas` Docker 볼륨 세분화 스냅샷 및 클론 복제와 같이 FlexVol에서 제공되는 추가 기능 때문에 드라이버가 선호되는 NAS 솔루션입니다.

FlexVol에 허용 가능한 것보다 더 많은 Docker 볼륨이 필요한 경우 ONTAP-NAS-이코노미 또는 ONTAP-SAN-이코노미 드라이버를 선택합니다.

`ontap-nas-economy`드라이버는 자동으로 관리되는 FlexVol 볼륨 풀 내에서 Docker 볼륨을 ONTAP Qtree로 생성합니다. qtree는 일부 기능을 희생하여 클러스터 노드당 최대 100,000 및 클러스터당 2,400,000까지 훨씬 더 뛰어난 확장을 제공합니다. `ontap-nas-economy`드라이버는 Docker 볼륨 단위의 세분화된 스냅샷 또는 클론 복제를 지원하지 않습니다.

참고 Docker Swarm은 여러 노드 간의 볼륨 생성을 오케스트레이션하지 않으므로 이 ontap-nas-economy 드라이버는 현재 Docker Swarm에서 지원되지 않습니다.

`ontap-san-economy`드라이버는 자동으로 관리되는 FlexVol 볼륨의 공유 풀 내에 Docker 볼륨을 ONTAP LUN으로 생성합니다. 이렇게 하면 각 FlexVol가 하나의 LUN에만 제한되지 않으며 SAN 워크로드에 더 나은 확장성을 제공합니다. 스토리지 시스템에 따라 ONTAP는 클러스터당 최대 16384개의 LUN을 지원합니다. 볼륨이 그 아래에 LUN이 있으므로 이 드라이버는 Docker 볼륨 세부 스냅샷 및 클론 복제를 지원합니다.

`ontap-nas-flexgroup`병렬 처리 수를 수십 억 개의 파일로 구성된 페타바이트 범위로 확장할 수 있는 단일 볼륨으로 늘리려면 드라이버를 선택하십시오. FlexGroups의 이상적인 사용 사례로는 AI/ML/DL, 빅데이터 및 분석, 소프트웨어 빌드, 스트리밍, 파일 저장소 등이 있습니다. Trident는 FlexGroup 볼륨을 프로비저닝할 때 SVM에 할당된 모든 애그리게이트를 사용합니다. Trident의 FlexGroup 지원에도 다음과 같은 고려 사항이 있습니다.

- ONTAP 버전 9.2 이상이 필요합니다.
- 이번 작부터 FlexGroups는 NFS v3만 지원합니다.
- SVM에 대해 64비트 NFSv3 식별자를 사용하는 것이 좋습니다.
- 권장되는 최소 FlexGroup 멤버/볼륨 크기는 100GiB입니다.
- FlexGroup 볼륨에는 클론 복제가 지원되지 않습니다.

FlexGroups 및 FlexGroups에 적합한 워크로드에 대한 자세한 내용은 "NetApp FlexGroup 볼륨 모범 사례 및 구축 가이드 를 참조하십시오"을 참조하십시오.

동일한 환경에서 고급 기능과 대규모 확장을 사용하려면 Docker Volume Plugin의 여러 인스턴스를 를 사용하고 을 사용하여 ontap-nas-economy 인스턴스를 하나씩 실행할 수 ontap-nas 있습니다.

Trident에 대한 사용자 지정 ONTAP 역할

Privileges에서 작업을 수행할 때 ONTAP 관리자 역할을 사용할 필요가 없도록 최소 Trident로 ONTAP 클러스터 역할을 생성할 수 있습니다. Trident 백엔드 구성에 사용자 이름을 포함하면 Trident은 사용자가 생성한 ONTAP 클러스터 역할을 사용하여 작업을 수행합니다.

Trident 사용자 지정 역할 생성에 대한 자세한 내용은 을 "Trident 사용자 지정 역할 생성기"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 사용자에 대한 사용 이름 만들기:

```
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. 역할을 사용자에게 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 레벨에서 사용자 지정 역할을 생성하려면 * 클러스터 > 설정 * 을 선택합니다.

      SVM 레벨에서 사용자 지정 역할을 생성하려면 * 스토리지 > 스토리지 VM >> 설정 > 사용자 및 역할 * 을 선택합니다 `required SVM`.

   b. 사용자 및 역할 * 옆의 화살표 아이콘(*→ *)을 선택합니다.
   c. 역할 * 아래에서 * + 추가 * 를 선택합니다.
   d. 역할에 대한 규칙을 정의하고 * 저장 * 을 클릭합니다.
2. * 역할을 Trident 사용자에게 매핑 *: + * 사용자 및 역할 * 페이지에서 다음 단계를 수행하십시오.
   a. 사용자 * 아래에서 추가 아이콘 * + * 를 선택합니다.
   b. 필요한 사용자 이름을 선택하고 * Role * 에 대한 드롭다운 메뉴에서 역할을 선택합니다.
   c. 저장 * 을 클릭합니다.

자세한 내용은 다음 페이지를 참조하십시오.

- "ONTAP 관리를 위한 사용자 지정 역할" 또는 "사용자 지정 역할을 정의합니다"
- "역할 및 사용자 작업"

ONTAP 구성 파일의 예

▼ 드라이버에 대한 NFS 예 <code>ontap-nas</code>

```
{
  "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"
  }
}
```

▼ 드라이버에 대한 NFS 예 <code>ontap-nas-flexgroup</code>

```
{
  "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"
  }
}
```

▼ 드라이버에 대한 NFS 예 <code>ontap-nas-economy</code>

```
{
  "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"
}
```

▼ 드라이버에 대한 **iSCSI** 예 **<code>ontap-san</code>**

```
{
  "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"
}
```

▼ 드라이버에 대한 **NFS** 예 **<code>ontap-san-economy</code>**

```
{
  "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"
}
```

▼ 드라이버에 대한 **NVMe/TCP** 예 **<code>ontap-san</code>**

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
```

```
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

▼ **<code> ONTAP-SAN </code>** 드라이버에 대한 **SCSI over FC** 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

Element 소프트웨어(NetApp HCI/SolidFire)를 사용하는 경우 글로벌 구성 값 외에도 이러한 옵션을 사용할 수 있습니다.

옵션을 선택합니다	설명	예
끝점	<a href="https://<login>:<password>@<mvip>/json-rpc/<element-version>" class="bare">https://<login>:<password>@<mvip>/json-rpc/<element-version>; 으로 문의하십시오	https://admin:admin@192.168.160.3/json-rpc/8.0 으로 문의하십시오
'VIP'	iSCSI IP 주소 및 포트	10.0.0.7:3260
테난트이름	사용할 SolidFireF 테넌트(찾을 수 없는 경우 생성됨)	`docker`
이니토IFace	iSCSI 트래픽을 기본 인터페이스가 아닌 인터페이스로 제한할 때 인터페이스를 지정합니다	`default`
'유형'	QoS 사양	아래 예를 참조하십시오

옵션을 선택합니다	설명	예
레가시NamePrefix	업그레이드된 Trident 설치의 접두사 1.3.2 이전 버전의 Trident를 사용하고 기존 볼륨으로 업그레이드를 수행하는 경우 볼륨 이름 방법을 통해 매핑된 이전 볼륨에 액세스하려면 이 값을 설정해야 합니다.	`netappdvp-`

졸idfire-san 드라이버는 Docker Swarm을 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

Trident를 Docker와 함께 사용할 때 알려진 문제 및 제한 사항에 대한 정보를 찾을 수 있습니다.

이전 버전에서 **Trident Docker** 볼륨 플러그인을 **20.10** 이상으로 업그레이드하면 해당 파일 또는 디렉터리 오류가 없는 업그레이드 오류가 발생합니다.

해결 방법

1. 플러그인을 비활성화합니다.

```
docker plugin disable -f netapp:latest
```

2. 플러그인을 제거합니다.

```
docker plugin rm -f netapp:latest
```

3. 추가 'config' 매개 변수를 제공하여 플러그인을 다시 설치합니다.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

볼륨 이름은 최소 **2**자 이상이어야 합니다.

참고	이는 Docker 클라이언트의 제한 사항입니다. 클라이언트는 단일 문자 이름을 Windows 경로로 해석합니다. "버그 25773을 참조하십시오".

Docker Swarm에는 모든 스토리지와 드라이버 조합에서 **Trident**이 이를 지원하지 못하는 특정한 동작이 있습니다.

- 현재 Docker Swarm은 볼륨 ID 대신 볼륨 이름을 고유한 볼륨 식별자로 사용합니다.
- 볼륨 요청은 Swarm 클러스터의 각 노드로 동시에 전송됩니다.
- 볼륨 플러그인(Trident 포함)은 Swarm 클러스터의 각 노드에서 독립적으로 실행되어야 합니다. ONTAP의 작동 방식 및 및 및 `ontap-san` 드라이버의 작동 방식 때문에 `ontap-nas` 이러한 제한 내에서 작동할 수 있는 것은 오직 한 가지뿐입니다.

나머지 드라이버는 경합 조건과 같이 명확한 "승자"가 없는 단일 요청에 대해 대량의 볼륨을 생성할 수 있는 문제가 발생합니다. 예를 들어 Element에는 볼륨이 동일한 이름을 갖지만 ID를 가질 수 있는 기능이 있습니다.

NetApp은 Docker 팀에 피드백을 제공했지만, 향후 소구에 대한 표시는 제공하지 않습니다.

FlexGroup를 프로비저닝하고 있는 경우, 두 번째 **FlexGroup**에 프로비저닝되는 **FlexGroup**와 공통되는 하나 이상의 애그리게이트가 있는 경우 **ONTAP**는 두 번째 **FlexGroup**를 프로비저닝하지 않습니다.

모범 사례 및 권장사항

Trident 배포 권장 사항

Trident를 구축할 때 여기에 나열된 권장 사항을 사용합니다.

전용 네임스페이스에 구축

"네임스페이스" 서로 다른 응용 프로그램 간의 관리 분리 기능을 제공하며 리소스 공유에 대한 장벽입니다. 예를 들어, 한 네임스페이스의 PVC는 다른 네임스페이스에서 사용할 수 없습니다. Trident는 Kubernetes 클러스터의 모든 네임스페이스에 PV 리소스를 제공하므로 Privileges가 상승한 서비스 계정을 활용합니다.

또한 Trident Pod에 액세스하면 사용자가 스토리지 시스템 자격 증명 및 기타 중요한 정보에 액세스할 수 있습니다. 애플리케이션 사용자 및 관리 애플리케이션에서 Trident 객체 정의 또는 POD 자체에 액세스할 수 없도록 하는 것이 중요합니다.

할당량 및 범위 제한을 사용하여 스토리지 사용량을 제어할 수 있습니다

Kubernetes에는 2가지 기능이 있으며, 이 기능을 조합하여 애플리케이션의 리소스 사용을 제한하는 강력한 메커니즘을 제공합니다. 를 클릭합니다 "스토리지 할당량 메커니즘" 관리자가 네임스페이스별로 글로벌 및 스토리지 클래스별, 용량 및 오브젝트 수 사용 제한을 구현할 수 있도록 지원 또한 를 사용합니다 "범위 제한" 요청이 프로비저닝 사용자에게 전달되기 전에 PVC 요청이 최소값 및 최대값 내에 있는지 확인합니다.

이러한 값은 네임스페이스 단위로 정의됩니다. 즉, 각 네임스페이스에는 리소스 요구 사항에 맞는 값이 정의되어 있어야 합니다. 에 대한 자세한 내용은 여기 를 참조하십시오 "할당량을 활용하는 방법".

Trident 스토리지 구성 모범 사례

NetApp 포트폴리오의 각 스토리지 플랫폼은 컨테이너식으로 애플리케이션에 이점을 제공하는 고유한 기능을 제공합니다.

플랫폼 개요

Trident는 ONTAP 및 요소와 함께 작동합니다. 모든 애플리케이션과 시나리오에 적합한 플랫폼이 한 개 있는 것은 아니지만, 플랫폼을 선택할 때 애플리케이션과 장치를 관리하는 팀의 요구 사항을 고려해야 합니다.

활용 중인 프로토콜을 사용하여 호스트 운영 체제의 기준 모범 사례를 따라야 합니다. 필요에 따라 특정 애플리케이션에 맞게 스토리지를 최적화할 수 있도록 백엔드, 스토리지 클래스 및 PVC 설정과 함께 사용 가능한 경우 애플리케이션 Best Practice를 통합하는 것을 고려할 수 있습니다.

ONTAP 및 Cloud Volumes ONTAP 모범 사례

Trident를 위한 ONTAP 및 Cloud Volumes ONTAP를 구성하기 위한 모범 사례에 대해 알아보십시오.

다음 권장 사항은 Trident에서 동적으로 프로비저닝되는 볼륨을 사용하는 컨테이너식 워크로드에 대한 ONTAP 구성 지침입니다. 각 항목을 고려하여 작업 환경의 적절성을 판단해야 합니다.

Trident 전용 SVM을 사용하십시오

SVM(스토리지 가상 시스템)은 ONTAP 시스템의 테넌트 간에 격리하고 관리를 제공합니다. SVM을 애플리케이션 전용으로 사용하면 권한을 위임하고 리소스 사용을 제한하는 모범 사례를 적용할 수 있습니다.

SVM 관리를 위해 몇 가지 옵션을 사용할 수 있습니다.

- 백엔드 구성에서 클러스터 관리 인터페이스를 적절한 자격 증명과 함께 제공하고 SVM 이름을 지정합니다.
- ONTAP System Manager 또는 CLI를 사용하여 SVM을 위한 전용 관리 인터페이스를 생성합니다.
- 관리 역할을 NFS 데이터 인터페이스와 공유합니다.

각 경우 인터페이스가 DNS에 있어야 하며, Trident를 구성할 때 DNS 이름을 사용해야 합니다. 이렇게 하면 네트워크 ID 보존을 사용하지 않고 SVM-DR과 같은 일부 DR 시나리오를 간편하게 수행할 수 있습니다.

SVM에 전용 또는 공유 관리 LIF를 사용하는 것이 더 이상 선호되지 않지만, 선택한 접근 방식에 맞게 네트워크 보안 정책을 조정해야 합니다. 관리 LIF는 DNS를 통해 액세스할 수 있어야 하며, 유연성을 극대화해야 합니다 "SVM-DR" Trident와 함께 사용합니다.

최대 볼륨 수를 제한합니다

ONTAP 스토리지 시스템의 최대 볼륨 수는 소프트웨어 버전과 하드웨어 플랫폼에 따라 다릅니다. 을 참조하십시오 "NetApp Hardware Universe를 참조하십시오" 정확한 제한을 결정하는 특정 플랫폼 및 ONTAP 버전. 볼륨 수가 소진되면 프로비저닝 작업이 Trident뿐 아니라 모든 스토리지 요청에 대해 실패합니다.

Trident의 'ONTAP-NAS' 및 'ONTAP-SAN' 드라이버는 생성되는 각 Kubernetes 영구 볼륨(PV)에 대해 FlexVolume을 프로비저닝합니다. 'ONTAP-NAS-이코노미' 드라이버는 200개의 PVS에 대해 약 1개의 FlexVolume을 생성합니다(50개에서 300개로 구성 가능). 'ONTAP-SAN-이코노미' 드라이버는 PVS 100대당 약 1개의 FlexVolume을 생성합니다(50개에서 200개로 구성 가능). Trident가 스토리지 시스템에서 사용 가능한 모든 볼륨을 사용하지 않도록 하려면 SVM에 제한을 설정해야 합니다. 이 작업은 명령줄에서 수행할 수 있습니다.

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

'최대 볼륨'의 값은 환경에 따라 몇 가지 기준에 따라 달라집니다.

- ONTAP 클러스터에 있는 기존 볼륨의 수입니다
- 다른 애플리케이션에 대해 Trident 외부에 프로비저닝할 것으로 예상되는 볼륨 수입니다
- Kubernetes 애플리케이션에서 사용할 것으로 예상되는 영구 볼륨의 수입니다

max-volumes 값은 개별 ONTAP 노드가 아닌 ONTAP 클러스터의 모든 노드에 프로비저닝된 총 볼륨입니다. 결과적으로, ONTAP 클러스터 노드에 다른 노드보다 훨씬 더 많은 Trident 프로비저닝 볼륨이 있을 수 있는 몇 가지 조건이 발생할 수 있습니다.

예를 들어, 2노드 ONTAP 클러스터에는 최대 2000개의 FlexVol 볼륨을 호스팅할 수 있습니다. 최대 볼륨 수를 1250으로 설정하면 매우 적절합니다. 그러나 한 노드에서만 SVM에 할당되거나 한 노드에서 할당된 애그리게이트를 용량 등)에 대해 프로비저닝할 수 없는 경우 "애그리게이트" 다른 노드는 모든 Trident 프로비저닝된 볼륨의 대상이 됩니다. 즉, 값에 도달하기 전에 해당 노드의 볼륨 제한에 도달하여 Trident 및 해당 노드를 사용하는 기타 볼륨 작업에 모두 영향을 줄 수 있습니다 `max-volumes`. * 클러스터의 각 노드에서 애그리게이트가 동일한 수의 Trident가 사용하는 SVM에 할당되도록 하면 이러한 상황을 방지할 수 있습니다. *

볼륨의 클론을 생성합니다

NetApp Trident 다음을 사용할 때 볼륨 복제를 지원합니다. `ontap-nas` , `ontap-san` , 그리고 `solidfire-san` 스토리지 드라이버. 사용 시 `ontap-nas-flexgroup` 또는 `ontap-nas-economy` 드라이버, 복제가 지원되지 않습니다. 기존 볼륨에서 새 볼륨을 생성하면 새로운 스냅샷이 생성됩니다.

경고	다른 StorageClass와 연결된 PVC를 복제하지 마십시오. 호환성을 보장하고 예기치 않은 동작을 방지하려면 동일한 StorageClass 내에서 복제 작업을 수행하십시오.

Trident에서 생성한 볼륨의 최대 크기를 제한합니다

Trident에서 생성할 수 있는 볼륨의 최대 크기를 구성하려면 'backend.json' 정의에 있는 'limitVolumeSize' 매개 변수를 사용하십시오.

스토리지 어레이에서 볼륨 크기를 제어하는 것 외에도 Kubernetes 기능을 활용해야 합니다.

Trident에 의해 생성되는 **FlexVols**의 최대 크기를 제한합니다

ONTAP-SAN-Economy 및 ONTAP-NAS-Economy 드라이버의 풀로 사용되는 FlexVol의 최대 크기를 구성하려면 `limitVolumePoolSize` `backend.json` 정의에 매개 변수를 사용하십시오.

양방향 **CHAP**를 사용하도록 **Trident**를 구성합니다

백엔드 정의에 CHAP 이니시에이터와 타겟 사용자 이름 및 암호를 지정하고 SVM에서 Trident가 CHAP를 사용하도록 설정할 수 있습니다. 를 사용합니다 `useCHAP` 매개 변수 백엔드 구성에서 Trident는 CHAP를 사용하여 ONTAP 백엔드에 대한 iSCSI 연결을 인증합니다.

SVM QoS 정책을 생성하고 사용합니다

SVM에 적용되는 ONTAP QoS 정책을 활용하여 Trident에서 프로비저닝된 볼륨에서 사용할 수 있는 IOPS 수를 제한합니다. 그러면 에 도움이 됩니다 "괴롭힘을 방지합니다" 또는 Trident SVM 외부의 워크로드에 영향을 주지 않는 제어 컨테이너

몇 가지 단계로 SVM에 대한 QoS 정책을 생성할 수 있습니다. 가장 정확한 정보는 사용 중인 ONTAP 버전 설명서를 참조하십시오. 아래 예는 SVM에 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 사용하는 ONTAP 버전에서 지원하는 경우에는 최소 QoS를 사용하여 컨테이너화된 워크로드에 대한 처리량을 보장하는 것을 고려할 수 있습니다. 적응형 QoS는 SVM 레벨 정책과 호환되지 않습니다.

컨테이너화된 워크로드 전용 IOPS 수는 다양한 측면에 따라 다릅니다. 그 밖의 다른 사항으로는 다음과 같은 것들이 있습니다.

- 기타 워크로드는 스토리지 어레이를 사용합니다. 스토리지 리소스를 활용하여 Kubernetes 구축과 관련되지 않은 다른 워크로드가 있는 경우, 해당 워크로드가 실수로 영향을 받지 않도록 주의해야 합니다.
- 컨테이너에서 실행 중인 예상 워크로드 IOPS 요구사항이 높은 워크로드를 컨테이너에서 실행할 경우 QoS 정책이 낮으면 잘못된 경험이 될 수 있습니다.

SVM 레벨에서 할당된 QoS 정책을 사용하면 동일한 IOPS 풀을 공유하는 SVM에 프로비저닝된 모든 볼륨이 생성된다는 점을 기억해야 합니다. 컨테이너화된 애플리케이션 중 하나 또는 그 수가 적은 경우 높은 IOPS 요구사항이 있으면 다른 컨테이너화된 워크로드에 문제가 될 수 있습니다. 이 경우 외부 자동화를 사용하여 볼륨당 QoS 정책을 할당하는 것을 고려할 수 있습니다.

중요함	ONTAP 버전이 9.8 이전인 경우 SVM * 에만 QoS 정책 그룹을 할당해야 합니다.

Trident에 대한 **QoS** 정책 그룹을 생성합니다

QoS(서비스 품질)는 경쟁 워크로드로부터 주요 워크로드의 성능이 저하되지 않도록 보장합니다. ONTAP QoS 정책 그룹은 볼륨에 대한 QoS 옵션을 제공하고 사용자가 하나 이상의 워크로드에 대한 처리량 한도를 정의할 수 있도록 지원합니다. QoS에 대한 자세한 내용은 를 참조하십시오 "QoS를 통해 처리량 보장".
백엔드에서 또는 스토리지 풀에 QoS 정책 그룹을 지정할 수 있으며, 이러한 그룹은 해당 풀 또는 백엔드에서 생성된 각 볼륨에 적용됩니다.

ONTAP에는 기존 QoS 정책과 적응형 서비스 두 가지 QoS 정책 그룹이 있습니다. 기존 정책 그룹은 IOPS 단위로 최대 또는 최소 단위의 고정 처리량을 제공합니다. 적응형 QoS는 워크로드 크기에 따라 처리량을 자동으로 확장하므로 워크로드 크기에 따라 IOPS와 TB|GB의 비율을 유지합니다. 따라서 대규모 구축 환경에서 수백 또는 수천 개의 워크로드를 관리할 경우 상당한 이점이 있습니다.

QoS 정책 그룹을 생성할 때는 다음 사항을 고려하십시오.

- 백엔드 구성의 "deefaults" 블록에 qosPolicy 키를 설정해야 합니다. 다음 백엔드 구성 예를 참조하십시오.

```
---
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
```

- 각 볼륨이 정책 그룹에서 지정한 전체 처리량을 얻을 수 있도록 볼륨별로 정책 그룹을 적용해야 합니다. 공유 정책 그룹은 지원되지 않습니다.

QoS 정책 그룹에 대한 자세한 내용은 을 "ONTAP 명령 참조입니다"참조하십시오.

스토리지 리소스에 대한 액세스 권한을 **Kubernetes** 클러스터 구성원으로 제한합니다

Trident에서 생성한 NFS 볼륨, iSCSI LUN 및 FC LUN에 대한 액세스를 제한하는 것은 Kubernetes 배포를 위한 보안 상태의 중요 구성 요소입니다. 이렇게 하면 Kubernetes 클러스터의 일부가 아닌 호스트가 볼륨에 액세스하고 예기치 않게 데이터를 수정하는 것을 방지할 수 있습니다.

네임스페이스가 Kubernetes의 리소스에 대한 논리적 경계라는 것을 이해하는 것이 중요합니다. 동일한 네임스페이스의 리소스를 공유할 수 있다고 가정하지만, 특히 상호 네임스페이스 기능이 없다는 것이 중요합니다. 즉, PVS는 글로벌 객체이지만 PVC에 바인딩되면 동일한 네임스페이스에 있는 Pod에서만 액세스할 수 있습니다. * 적절한 경우 네임스페이스를 사용하여 구분을 제공하는 것이 중요합니다. *

Kubernetes 컨텍스트에서 데이터 보안과 관련하여 대부분의 조직은 컨테이너 내의 프로세스가 호스트에 마운트된 스토리지에 액세스할 수 있지만 컨테이너용 프로세스는 아닙니다. "네임스페이스" 이러한 유형의 손상을 방지하도록 설계되었습니다. 그러나 권한 있는 컨테이너에는 한 가지 예외가 있습니다.

권한 있는 컨테이너는 일반적인 것보다 훨씬 더 많은 호스트 수준 권한으로 실행되는 컨테이너입니다. 이러한 기능은 기본적으로 거부되지 않으므로 을 사용하여 기능을 사용하지 않도록 설정해야 합니다 "POD 보안 정책".

Kubernetes 및 외부 호스트 모두에서 액세스가 필요한 볼륨의 경우, Trident에서 관리하지 않고 관리자가 PV를 도입한 상태로 스토리지를 기존 방식으로 관리해야 합니다. 이렇게 하면 Kubernetes 및 외부 호스트의 연결이 모두 끊기고 볼륨을 더 이상 사용하지 않는 경우에만 스토리지 볼륨이 폐기됩니다. 또한, 맞춤형 엑스포트 정책을 적용하여 Kubernetes 클러스터 노드 및 Kubernetes 클러스터 외부의 타겟 서버에서 액세스할 수 있습니다.

전용 인프라 노드(예: OpenShift) 또는 사용자 애플리케이션을 예약할 수 없는 다른 노드를 구축하는 경우, 별도의 엑스포트 정책을 사용하여 스토리지 리소스에 대한 액세스를 더욱 제한해야 합니다. 여기에는 해당 인프라 노드에 배포된 서비스(예: OpenShift Metrics 및 Logging 서비스)에 대한 엑스포트 정책과 비인프라 노드에 배포되는 표준 애플리케이션이 포함됩니다.

전용 엑스포트 정책을 사용하십시오

Kubernetes 클러스터에 있는 노드에만 액세스할 수 있도록 각 백엔드에 대한 엑스포트 정책이 있어야 합니다. Trident는 엑스포트 정책을 자동으로 생성하고 관리할 수 있습니다. 이러한 방법으로 Trident는 Kubernetes 클러스터의 노드에 프로비저닝되는 볼륨에 대한 액세스를 제한하고 노드 추가/삭제를 단순화합니다.

또는 수동으로 엑스포트 정책을 생성하여 각 노드 액세스 요청을 처리하는 하나 이상의 엑스포트 규칙으로 채울 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.

사용 안 함 `showmount` 애플리케이션 **SVM**을 위해

이 `showmount` 기능을 사용하면 NFS 클라이언트가 SVM에서 사용 가능한 NFS 엑스포트 목록을 쿼리할 수 있습니다. Kubernetes 클러스터에 배포된 Pod는 에 대해 명령을 실행하고 액세스 권한이 없는 마운트를 포함하여 사용 가능한 마운트 목록을 받을 수 있습니다 `showmount -e`. 이는 그 자체로 보안 문제가 아니라, 권한이 없는 사용자가 NFS

내보내기에 연결하는 데 도움이 될 수 있는 불필요한 정보를 제공합니다.

SVM 레벨의 ONTAP CLI 명령을 사용하여 'howmount'를 비활성화해야 합니다.

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 모범 사례

Trident를 위한 SolidFire 스토리지를 구성하기 위한 모범 사례에 대해 알아보십시오.

SolidFire 계정을 만듭니다

각 SolidFire 계정은 고유한 볼륨 소유자를 나타내며 자체 CHAP(Challenge-Handshake 인증 프로토콜) 자격 증명을 받습니다. 계정 이름 및 상대 CHAP 자격 증명을 사용하거나 볼륨 액세스 그룹을 통해 계정에 할당된 볼륨에 액세스할 수 있습니다. 계정에는 최대 2천 개의 볼륨이 할당될 수 있지만 볼륨은 하나의 계정에만 속할 수 있습니다.

QoS 정책을 생성합니다

여러 볼륨에 적용할 수 있는 표준화된 서비스 품질 설정을 만들어 저장하려면 SolidFire 서비스 품질(QoS) 정책을 사용하십시오.

볼륨별로 QoS 매개 변수를 설정할 수 있습니다. QoS를 정의하는 세 가지 구성 가능한 매개 변수, 즉 Min IOPS, Max IOPS, Burst IOPS를 설정하여 각 볼륨의 성능을 보장할 수 있습니다.

4KB 블록 크기에 대해 가능한 최소, 최대 및 버스트 IOPS 값입니다.

IOPS 매개 변수입니다	정의	최소 값	기본값	최대 가치**(4KB)**
최소 IOPS	볼륨에 대한 보장된 성능 수준.	50	50	15000
최대 IOPS	성능은 이 제한을 초과하지 않습니다.	50	15000	200,000
버스트 IOPS	짧은 버스트 시나리오에서 허용되는 최대 IOPS입니다.	50	15000	200,000

참고 최대 IOPS와 버스트 IOPS는 최대 200,000으로 설정할 수 있지만, 실제 볼륨의 최대 성능은 클러스터 사용량 및 노드당 성능에 의해 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

요소는 CHAP 및 vag(볼륨 액세스 그룹)의 두 가지 인증 방법을 지원합니다. CHAP는 CHAP 프로토콜을 사용하여 호스트를 백엔드에 인증합니다. 볼륨 액세스 그룹은 프로비전되는 볼륨에 대한 액세스를 제어합니다. NetApp은

CHAP를 사용하여 인증을 수행하는 것이 더 간단하고 확장 제한이 없기 때문에 CHAP를 사용하는 것이 좋습니다.

참고	CSI 프로비저닝이 강화된 Trident는 CHAP 인증 사용을 지원합니다. VAG는 일반적인 비 CSI 작동 모드에서만 사용해야 합니다.

CHAP 인증(이니시에이터가 대상 볼륨 사용자인지 확인)은 계정 기반 액세스 제어에서만 지원됩니다. CHAP를 인증에 사용하는 경우 단방향 CHAP 및 양방향 CHAP의 두 가지 옵션을 사용할 수 있습니다. 단방향 CHAP는 SolidFire 계정 이름 및 이니시에이터 암호를 사용하여 볼륨 액세스를 인증합니다. 양방향 CHAP 옵션은 볼륨이 계정 이름과 이니시에이터 암호를 통해 호스트를 인증한 다음 호스트가 계정 이름과 타겟 암호를 통해 볼륨을 인증하기 때문에 볼륨을 인증하는 가장 안전한 방법을 제공합니다.

그러나 CHAP를 설정할 수 없고 VAG가 필요한 경우 액세스 그룹을 생성하고 호스트 이니시에이터 및 볼륨을 액세스 그룹에 추가합니다. 액세스 그룹에 추가하는 각 IQN은 CHAP 인증을 사용하거나 사용하지 않고 그룹의 각 볼륨에 액세스할 수 있습니다. iSCSI 이니시에이터가 CHAP 인증을 사용하도록 구성된 경우 계정 기반 액세스 제어가 사용됩니다. iSCSI 초기자가 CHAP 인증을 사용하도록 구성되지 않은 경우 볼륨 액세스 그룹 액세스 제어가 사용됩니다.

자세한 정보는 어디서 찾을 수 있습니까?

다음은 몇 가지 모범 사례 문서입니다. 를 검색합니다 "NetApp 라이브러리" 최신 버전의 경우.

- ONTAP *
- "NFS Best Practice and Implementation Guide를 참조하십시오"
- "SAN 관리" (iSCSI용)
- "RHEL용 iSCSI Express 구성"

Element 소프트웨어 *

- "Linux용 SolidFire 구성"
- NetApp HCI *
- "NetApp HCI 구축 사전 요구 사항"
- "NetApp 배포 엔진에 액세스합니다"
- 응용 프로그램 모범 사례 정보 *
- "ONTAP 기반 MySQL의 모범 사례"
- "SolidFire 기반 MySQL의 모범 사례"
- "NetApp SolidFire 및 Cassandra"
- "SolidFire에 대한 Oracle 모범 사례"
- "SolidFire에 대한 PostgreSQL Best Practice"

모든 애플리케이션에 구체적인 지침이 있는 것은 아니며 NetApp 팀과 함께 을 사용하는 것이 중요합니다 "NetApp 라이브러리" 최신 설명서를 참조하십시오.

Trident를 Kubernetes와 통합하십시오

Trident를 통합하려면 드라이버 선택 및 배포, 스토리지 클래스 설계, 가상 풀 설계, PVC(영구적

볼륨 클레임)가 스토리지 프로비저닝, 볼륨 운영, Trident를 사용하는 OpenShift 서비스 배포에 미치는 영향 등 통합이 필요합니다.

운전자 선택 및 전개

스토리지 시스템에 사용할 백엔드 드라이버를 선택하고 구축합니다.

ONTAP 백엔드 드라이버

ONTAP 백엔드 드라이버는 사용된 프로토콜과 스토리지 시스템에서 볼륨을 프로비저닝하는 방법에 따라 다릅니다. 따라서 배포할 드라이버를 결정할 때는 신중하게 고려해야 합니다.

상위 레벨에서는 애플리케이션에 공유 스토리지가 필요한 구성 요소(동일한 PVC에 액세스하는 여러 Pod)가 있는 경우 NAS 기반 드라이버가 기본 선택이고 블록 기반 iSCSI 드라이버는 비공유 스토리지의 요구를 충족합니다. 애플리케이션의 요구사항 및 스토리지 및 인프라 팀의 편안함 수준을 기준으로 프로토콜을 선택합니다. 일반적으로 대부분의 애플리케이션에서 두 서버 간에 차이가 거의 없기 때문에 공유 스토리지(둘 이상의 POD에 동시 액세스가 필요한 경우)가 필요한지 여부에 따라 결정하는 경우가 많습니다.

사용 가능한 ONTAP 백엔드 드라이버는 다음과 같습니다.

- ONTAP-NAS: 프로비저닝되는 각 PV는 ONTAP FlexVolume입니다.
- ONTAP-NAS-이코노미: 각 PV 프로비저닝은 qtree이며 FlexVolume당 qtree(기본값 200)를 구성할 수 있습니다.
- ONTAP-NAS-flexgroup: 각 PV는 전체 ONTAP FlexGroup로 프로비저닝되고 SVM에 할당된 모든 애그리게이트가 사용됩니다.
- ONTAP-SAN: 각 PV는 자체 FlexVolume 내의 LUN입니다.
- ONTAP-SAN-이코노미: 각 PV는 LUN으로 프로비저닝되며 FlexVolume당 구성 가능한 LUN 수(기본값 100)가 있습니다.

세 개의 NAS 드라이버 중 하나를 선택할 경우 해당 기능에 약간의 영향을 줍니다. 이 기능은 응용 프로그램에서 사용할 수 있습니다.

아래 표에서 모든 기능이 Trident를 통해 표시되는 것은 아닙니다. 용량 할당 후 기능을 적용하려면 스토리지 관리자가 일부 기능을 적용해야 합니다. 위 첨자 각주는 기능 및 드라이버별 기능을 구별합니다.

ONTAP NAS 드라이버	스냅샷 수	복제	동적 엑스포트 정책	다중 연결	**QoS**를 참조하십시오	크기 조정	복제
'ONTAP-NAS'	예	예	Yesfootnote: 5[]	예	Yesfootnote: 1[]	예	Yesfootnote: 1[]
ONTAP-NAS-이코노미	메모:3[]	메모:3[]	Yesfootnote: 5[]	예	메모:3[]	예	메모:3[]
'ONTAP-NAS-Flexgroup'	Yesfootnote: 1[]	아니요	Yesfootnote: 5[]	예	Yesfootnote: 1[]	예	Yesfootnote: 1[]

Trident는 ONTAP용 SAN 드라이버 2개를 제공합니다. 이러한 드라이버는 다음과 같습니다.

ONTAP SAN 드라이버	스냅샷 수	복제	다중 연결	양방향 **CHAP**	**QoS**를 참조하십시오	크기 조정	복제
'ONTAP-SAN'	예	예	Yesfootnote: 4[]	예	Yesfootnote: 1[]	예	Yesfootnote: 1[]
ONTAP-SAN-이코노미	예	예	Yesfootnote: 4[]	예	메모:3[]	예	메모:3[]

> 위 표의 각주: Yes[1]: Trident에서 관리되지 않음 Yes[2]: Trident에서 관리하지만 PV 세밀한 NO[3]: Trident에서 관리하지 않으며 PV 세밀한 예각주:4[]: 원시 블록 볼륨에 대해 지원됨 예: 5[]: Trident에서 지원합니다

PV 세분화되지 않은 기능은 전체 FlexVolume에 적용되고 모든 PVS(즉, 공유 FlexVol의 qtree 또는 LUN)는 공통 스케줄을 공유합니다.

위 표에서 볼 수 있듯이, ONTAP-NAS와 ONTAP-NAS-이코노미 간의 기능 대부분은 동일합니다. 그러나 ONTAP-NAS-이코노미 드라이버는 PV 단위의 일정 제어 기능을 제한하므로 특히 재해 복구 및 백업 계획에 영향을 줄 수 있습니다. ONTAP 스토리지에서 PVC 클론 기능을 활용하고자 하는 개발팀은 ONTAP-NAS, ONTAP-SAN 또는 ONTAP-SAN 절약 드라이버를 사용할 때만 가능합니다.

참고	좋idfire-san의 드라이버도 PVC를 클로닝할 수 있습니다.

Cloud Volumes ONTAP 백엔드 드라이버

Cloud Volumes ONTAP은 NAS 및 SAN 프로토콜(NFS, SMB/CIFS 및 iSCSI)을 지원하는 파일 공유 및 블록 레벨 스토리지 등 다양한 활용 사례에 맞게 엔터프라이즈급 스토리지 기능과 함께 데이터 제어를 제공합니다. Cloud Volume ONTAP와 호환되는 드라이버는 ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-SAN, ONTAP-SAN 경제입니다. Cloud Volume ONTAP for Azure, Cloud Volume ONTAP for GCP에 적용할 수 있습니다.

ONTAP 백엔드 드라이버용 **Amazon FSx**

Amazon FSx for NetApp ONTAP를 사용하면 익숙한 NetApp 기능, 성능 및 관리 기능을 활용하면서 AWS에 데이터를 저장할 때의 단순성, 민첩성, 보안 및 확장성을 활용할 수 있습니다. FSx for ONTAP은 많은 ONTAP 파일 시스템 기능과 관리 API를 지원합니다. Cloud Volume ONTAP와 호환되는 드라이버는 입니다 `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` 및 `ontap-san-economy`.

NetApp HCI/SolidFire 백엔드 드라이버

NetApp HCI/SolidFire 플랫폼과 함께 사용되는 'olidfire-SAN' 드라이버는 관리자가 QoS 제한을 기반으로 Trident에 대한 Element 백엔드를 구성하는 데 도움이 됩니다. Trident에서 프로비저닝한 볼륨에 대한 특정 QoS 제한을 설정하기 위해 백엔드를 설계하려면 백엔드 파일에 'type' 매개 변수를 사용하십시오. 또한 관리자는 'limitVolumeSize' 매개 변수를 사용하여 스토리지에 생성할 수 있는 볼륨 크기를 제한할 수 있습니다. 현재 볼륨 크기 조정 및 볼륨 복제와 같은 Element 스토리지 기능은 'olidfire-SAN' 드라이버를 통해 지원되지 않습니다. 이러한 작업은 Element 소프트웨어 웹 UI를 통해 수동으로 수행해야 합니다.

SolidFire 드라이버	스냅샷 수	복제	다중 연결	**CHAP**	**QoS**를 참조하십시오	크기 조정	복제
'솔더불-산'	예	예	Yes[2]	예	예	예	Yesfootnote: 1[]

> 각주: Yes[1]: Trident에서 관리하지 않습니다. 예각주:2[]: 원시 블록 볼륨에 대해 지원됩니다

Azure NetApp Files 백엔드 드라이버

Trident는 `azure-netapp-files` 드라이버를 사용하여 "Azure NetApp Files" 서비스를 관리합니다.

이 드라이버에 대한 자세한 내용 및 구성 방법은 을 "Azure NetApp Files에 대한 Trident 백엔드 구성"참조하십시오.

Azure NetApp Files 드라이버	스냅샷 수	복제	다중 연결	**QoS**를 참조하십시오	비즈니스	복제
'Azure-NetApp-파일'	예	예	예	예	예	Yesfootnote: 1[]

> 각주: Yes[1]: Trident에서 관리하지 않습니다

Google Cloud NetApp Volumes 백엔드 드라이버

Trident는 NAS 워크로드용 `google-cloud-netapp-volumes` 드라이버와 SAN(iSCSI) 워크로드용 `google-cloud-netapp-volumes-san` 드라이버를 지원합니다. GCP에서 실행되는 OpenShift 클러스터의 경우, 두 드라이버 모두 Trident 26.10부터 Google Cloud Workload Identity Federation을 지원합니다. 여기에는 자체 관리형 OpenShift Container Platform(IPI 배포를 포함한 OCP) 및 OpenShift Dedicated(OSD)가 포함됩니다.

백엔드 구성 및 WIF 필수 조건에 대해서는 "Google Cloud NetApp Volumes 구성" 및 "SAN 워크로드를 위해 Google Cloud NetApp Volumes 구성"를 참조하십시오.

스토리지 클래스 설계

Kubernetes Storage Class 객체를 생성하려면 개별 스토리지 클래스를 구성 및 적용해야 합니다. 이 섹션에서는 애플리케이션에 대한 스토리지 클래스를 설계하는 방법에 대해 설명합니다.

특정 백엔드 활용도

특정 스토리지 클래스 객체 내에서 필터링을 사용하여 해당 스토리지 클래스에 사용할 스토리지 풀 또는 풀 세트를 결정할 수 있습니다. Storage Class에서 'toragePools', 'additionalStoragePools', 'excludeStoragePools' 등의 세 가지 필터를 설정할 수 있습니다.

`storagePools`매개 변수를 사용하면 지정된 속성과 일치하는 풀 세트로 스토리지를 제한할 수 있습니다. `additionalStoragePools`매개 변수는 `Trident`가 프로비저닝에 사용하는 풀 세트를 속성 및 매개 변수로 선택한 풀 세트와 함께 확장하는 데 `storagePools` 사용됩니다. 매개 변수만 사용하거나 둘 모두를 함께 사용하여 적절한 스토리지 풀 세트가 선택되었는지 확인할 수 있습니다.

excludeStoragePools 매개 변수는 속성과 일치하는 나열된 풀 세트를 명시적으로 제외하는 데 사용됩니다.

QoS 정책을 에뮬레이트합니다

서비스 품질 정책을 에뮬레이트하기 위해 스토리지 클래스를 설계하려면 '미디어' 속성을 HDD 또는 'SSD'로 사용하여 스토리지 클래스를 생성합니다. 스토리지 클래스에 언급된 미디어 특성에 따라, Trident는 미디어 속성과 일치하도록 HDD 또는 SSD 애그리게이트를 제공하는 적절한 백엔드를 선택한 다음 볼륨 프로비저닝을 특정 애그리게이트로 전달합니다. 따라서 프리미엄 QoS 정책으로 분류될 수 있는 '미디어' 속성이 'SD'로 설정된 스토리지 클래스 Premium을 생성할 수 있습니다. 표준 QoS 정책으로 분류될 수 있는 미디어 속성을 'HDD'로 설정하는 또 다른 스토리지 클래스 표준을 생성할 수 있습니다. 또한 스토리지 클래스에서 ""IOPS"" 속성을 사용하여 QoS 정책으로 정의할 수 있는 Element 어플라이언스로 프로비저닝을 리디렉션할 수도 있습니다.

특정 기능을 기반으로 백엔드를 활용합니다

스토리지 클래스는 씬 및 일반 프로비저닝, 스냅샷, 클론 및 암호화와 같은 기능이 설정된 특정 백엔드에서 볼륨 프로비저닝을 수행하도록 설계되었습니다. 사용할 스토리지를 지정하려면 필요한 기능이 설정된 적절한 백엔드를 지정하는 스토리지 클래스를 생성합니다.

가상 풀

모든 Trident 백엔드에서 가상 풀을 사용할 수 있습니다. Trident에서 제공하는 모든 드라이버를 사용하여 백엔드에 대한 가상 풀을 정의할 수 있습니다.

가상 풀을 사용하면 관리자가 저장소 클래스를 통해 참조할 수 있는 백엔드에 대한 추상화 수준을 생성하여 백엔드에 볼륨을 보다 유연하고 효율적으로 배치할 수 있습니다. 동일한 서비스 클래스로 다른 백엔드를 정의할 수 있습니다. 또한 동일한 백엔드에서 여러 스토리지 풀을 생성할 수 있지만 특성이 다릅니다. 특정 레이블이 있는 선택기를 사용하여 스토리지 클래스를 구성하면 Trident에서 볼륨을 배치할 모든 선택기 레이블과 일치하는 백엔드를 선택합니다. 스토리지 클래스 선택기 레이블이 여러 스토리지 풀과 일치하는 경우 Trident는 볼륨 용량 할당을 위해 둘 중 하나를 선택합니다.

가상 풀 설계

백엔드를 생성할 때 일반적으로 매개변수 집합을 지정할 수 있습니다. 이전에는 관리자가 동일한 스토리지 자격 증명과 다른 매개변수 집합을 사용하여 다른 백엔드를 생성하는 것이 불가능했습니다. 가상 풀이 도입되면서 이 문제가 해결되었습니다. 가상 풀은 백엔드와 Kubernetes 스토리지 클래스 사이에 도입된 레벨 추상화로, 관리자가 백엔드에 관계없이 Kubernetes 스토리지 클래스를 통해 선택기로 참조할 수 있는 레이블과 함께 매개변수를 정의할 수 있도록 합니다. 가상 풀은 Trident 를 통해 지원되는 모든 NetApp 백엔드에 대해 정의할 수 있습니다. 해당 목록에는 SolidFire/NetApp HCI, ONTAP, Azure NetApp Files 포함됩니다.

참고	가상 풀을 정의할 때는 백엔드 정의에서 기존 가상 풀의 순서를 재정렬하지 않는 것이 좋습니다. 또한 기존 가상 풀의 속성을 편집/수정하고 대신 새 가상 풀을 정의하는 것이 좋습니다.

다양한 서비스 수준**/QoS** 에뮬레이션

서비스 클래스를 에뮬레이트하기 위한 가상 풀을 설계할 수 있습니다. Azure NetApp Files용 Cloud Volume Service에 대한 가상 풀 구현을 사용하여 다양한 서비스 클래스를 설정하는 방법을 살펴보겠습니다. 다양한 성능 수준을 나타내는 여러 레이블을 사용하여 Azure NetApp Files 백엔드를 구성합니다. 설정 `servicelevel` 적절한 성과 수준에 맞게 종횡비를 지정하고 각 레이블 아래에 다른 필요한 요소를 추가합니다. 이제 다른 가상 풀에 매핑할 다른 Kubernetes 스토리지 클래스를 생성합니다. 를 사용합니다 `parameters.selector` 필드에서 각 StorageClass는 볼륨을 호스팅하는 데 사용할 수 있는 가상 풀을 호출합니다.

특정 측면 지정

특정 측면의 여러 가상 풀을 단일 스토리지 백엔드에서 설계할 수 있습니다. 이를 위해 백엔드에 여러 레이블을 구성하고 각 레이블 아래에 필요한 측면을 설정합니다. 이제 를 사용하여 다양한 Kubernetes Storage 클래스를 생성할 수 있습니다 `parameters.selector` 다른 가상 풀에 매핑될 필드입니다. 백엔드에서 프로비저닝되는 볼륨에는 선택한 가상 풀에 정의된 측면이 있습니다.

스토리지 프로비저닝에 영향을 미치는 **PVC** 특성

요청된 스토리지 클래스를 벗어난 일부 매개변수는 PVC 생성 시 Trident 프로비저닝 결정 프로세스에 영향을 미칠 수 있습니다.

액세스 모드

PVC를 통한 저장 요청 시 필수 필드 중 하나가 액세스 모드입니다. 원하는 모드는 스토리지 요청을 호스팅하기 위해 선택한 백엔드에 영향을 줄 수 있습니다.

Trident는 다음 매트릭스에 따라 지정된 액세스 방법과 함께 사용되는 스토리지 프로토콜을 일치시키려고 시도합니다. 이는 기본 스토리지 플랫폼과 무관합니다.

	ReadWriteOnce 를 참조하십시오	**ReadOnlyMany** 를 참조하십시오	**ReadWriteMany** 를 참조하십시오
iSCSI	예	예	예(원시 블록)
NFS 를 참조하십시오	예	예	예

NFS 백엔드가 구성되지 않은 상태로 Trident 배포에 제출된 ReadWriteMany PVC에 대한 요청은 볼륨이 프로비저닝되지 않습니다. 이러한 이유로 요청자는 자신의 응용 프로그램에 적합한 액세스 모드를 사용해야 합니다.

볼륨 작업입니다

영구 볼륨 수정

영구 볼륨은 Kubernetes에서 두 가지 예외, 영구적 객체입니다. 생성된 후에는 부가세 반환 청구액 정책 및 크기를 수정할 수 있습니다. 그러나 이렇게 해도 볼륨의 일부 측면이 Kubernetes 외부에서 수정되는 것을 방지할 수 없습니다. 특정 애플리케이션에 맞게 볼륨을 사용자 지정하거나, 실수로 용량이 소비되지 않도록 하거나, 어떠한 이유로든 볼륨을 다른 스토리지 컨트롤러로 이동하는 것이 좋을 수 있습니다.

참고	Kubernetes 트리 내 프로비저닝은 현재 NFS, iSCSI 또는 FC PVS에 대한 볼륨 크기 조정 작업을 지원하지 않습니다. Trident은 NFS, iSCSI 및 FC 볼륨의 확장을 지원합니다.

PV의 접속 세부 정보는 생성 후 수정할 수 없습니다.

주문형 볼륨 스냅샷을 생성합니다

Trident는 CSI 프레임워크를 사용하여 주문형 볼륨 스냅샷 생성 및 스냅샷으로부터 PVC 생성을 지원합니다. 스냅샷은 편리한 데이터 시점 복사본을 유지 관리하는 방법을 제공하며 Kubernetes의 소스 PV와 독립적인 라이프사이클을 갖고 있습니다. 이러한 스냅샷을 사용하여 PVC를 복제할 수 있습니다.

스냅샷으로부터 볼륨을 생성합니다

Trident는 또한 볼륨 스냅샷에서 PersistentVolumes 생성을 지원합니다. 이를 위해 PersistentVolumeClaim을 생성하고 볼륨을 생성해야 하는 필수 스냅샷으로 을 언급하면 `datasource` 됩니다. Trident는 스냅샷에 데이터가 있는 볼륨을 생성하여 이 PVC를 처리합니다. 이 기능을 사용하면 지역 간에 데이터를 복제하거나 테스트 환경을 생성하거나 손상되거나 손상된 운영 볼륨을 전체적으로 교체하거나 특정 파일 및 디렉토리를 검색하여 연결된 다른 볼륨으로 전송할 수 있습니다.

클러스터에서 볼륨 이동

스토리지 관리자는 ONTAP 클러스터의 Aggregate와 컨트롤러 간에 볼륨을 스토리지 소비자로 중단 없이 이동할 수 있습니다. 대상 애그리게이트가 Trident가 사용 중인 SVM에 액세스할 수 있는 경우에만 이 작업이 Trident 또는 Kubernetes 클러스터에는 영향을 미치지 않습니다. 중요한 것은, aggregate가 SVM에 새로 추가된 경우 이를 Trident에 다시 추가하여 백엔드를 새로 고쳐야 한다는 것입니다. 이 경우 Trident가 SVM의 인벤토리를 다시 작성하도록 트리거하여 새 애그리게이트가 인식됩니다.

하지만 Trident에서는 백엔드 볼륨 이동이 자동으로 지원되지 않습니다. 여기에는 동일한 클러스터 내, 클러스터 간 또는 다른 스토리지 플랫폼(스토리지 시스템이 Trident에 연결된 SVM인 경우에도)에 있는 SVM이 포함됩니다.

볼륨이 다른 위치에 복사되면 볼륨 가져오기 기능을 사용하여 현재 볼륨을 Trident로 가져올 수 있습니다.

볼륨 확장

Trident NFS, iSCSI, FC PV의 크기 조정을 지원합니다. 이를 통해 사용자는 Kubernetes 계층을 통해 볼륨 크기를 직접 조정할 수 있습니다. ONTAP, SolidFire/ NetApp HCI 백엔드를 포함한 모든 주요 NetApp 스토리지 플랫폼에서 볼륨 확장이 가능합니다. 나중에 확장이 가능하도록 설정하세요. `allowVolumeExpansion` 에게 `true` 볼륨과 연결된 StorageClass에서. 영구 볼륨의 크기를 조정해야 할 때마다 다음을 편집합니다. `spec.resources.requests.storage` 영구 볼륨 클레임에 필요한 볼륨 크기에 대한 주석을 추가합니다. Trident 스토리지 클러스터의 볼륨 크기를 자동으로 조정합니다.

기존 볼륨을 **Kubernetes**로 임포트

볼륨 가져오기는 기존 스토리지 볼륨을 Kubernetes 환경으로 가져오는 기능을 제공합니다. 이는 현재 다음에서 지원됩니다. `ontap-nas` , `ontap-nas-flexgroup` , `solidfire-san` , 그리고 `azure-netapp-files` 운전자. 이 기능은 기존 애플리케이션을 Kubernetes로 이식할 때나 재해 복구 시나리오에서 유용합니다.

ONTAP 및 드라이버를 사용할 `solidfire-san` 경우 명령을 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` 사용하여 기존 볼륨을 Kubernetes로 임포트하여 Trident에서 관리할 수 있습니다. 가져오기 볼륨 명령에 사용된 PVC YAML 또는 JSON 파일은 Trident를 프로비저닝자로 식별하는 저장소 클래스를 가리킵니다. NetApp HCI/SolidFire 백엔드를 사용할 경우 볼륨 이름이 고유한지 확인합니다. 볼륨 이름이 중복되면 볼륨을 고유한 이름으로 복제하여 볼륨 가져오기 기능에서 볼륨 이름을 구분할 수 있도록 합니다.

만약 `azure-netapp-files` 드라이버가 사용되면 명령을 사용하세요 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 볼륨을 Kubernetes로 가져와서 Trident 에서 관리하도록 합니다. 이를 통해 고유한 볼륨 참조가 보장됩니다.

위의 명령을 실행하면 Trident는 백엔드에서 볼륨을 찾고 크기를 읽습니다. 구성된 PVC 볼륨 크기를 자동으로 추가(필요한 경우 덮어쓰기)합니다. 그런 다음 Trident는 새 PV를 만들고 Kubernetes는 PV에 PVC를 바인딩합니다.

특정 가져온 PVC가 필요한 컨테이너를 배포한 경우 PVC/PV 쌍이 볼륨 가져오기 프로세스를 통해 바인딩될 때까지 보류 상태로 유지됩니다. PVC/PV 쌍이 바인딩되면 다른 문제가 없는 한 컨테이너가 나타나야 합니다.

레지스트리 서비스

레지스트리의 스토리지 배포 및 관리는 에 설명되어 있습니다 "NetApp.IO를 참조하십시오" 에 있습니다 "블로그".

로깅 서비스

다른 OpenShift 서비스와 마찬가지로 로깅 서비스는 Ansible을 사용하여 인벤토리 파일에서 제공하는 구성 매개 변수로 배포됩니다 호스트가 플레이북에 제공됩니다. OpenShift를 설치한 후 초기 OpenShift 설치 중에 로깅을 배포하고 로깅을 배포하는 두 가지 설치 방법이 제공됩니다.

주의	Red Hat OpenShift 버전 3.9를 기준으로 공식 문서는 데이터 손상 관련 우려 때문에 로깅 서비스에 NFS를 사용할 것을 권장합니다. 이는 제품에 대한 Red Hat 테스트를 기반으로 합니다. ONTAP NFS 서버에는 이러한 문제가 없으며 로깅 구축을 쉽게 되돌릴 수 있습니다. 궁극적으로, 로깅 서비스를 위한 프로토콜을 선택할 수 있습니다. 두 가지 모두 NetApp 플랫폼을 사용할 때 효과가 있으며 원할 경우 NFS를 피할 이유가 없습니다.

로깅 서비스에서 NFS를 사용하도록 선택한 경우 설치 관리자의 실패를 방지하려면 Ansible 변수 "openshift_enable_unsupported_configurations"를 "true"로 설정해야 합니다.

시작하십시오

로깅 서비스는 필요에 따라 두 애플리케이션 및 OpenShift 클러스터 자체의 핵심 운영에 구축할 수 있습니다. 작업 로깅을 배포하려는 경우 변수 "openshift_logging_use_ops"를 "true"로 지정하면 서비스의 인스턴스 두 개가 만들어집니다. 작업에 대한 로깅 인스턴스를 제어하는 변수에는 "ops"가 포함되어 있지만 응용 프로그램의 인스턴스는 그렇지 않습니다.

기본 서비스에서 올바른 스토리지를 활용하기 위해서는 배포 방법에 따라 Ansible 변수를 구성하는 것이 중요합니다. 각 배포 방법에 대한 옵션을 살펴보겠습니다.

참고	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 변수는 제공된 세부 정보를 사용하여 로깅 서비스에 대한 PV 및 PVC를 생성하는 Ansible 플레이북을 만듭니다. 이 방법은 OpenShift 설치 후 구성 요소 설치 플레이북을 사용하는 것보다 훨씬 덜 유연하지만, 기존 볼륨을 사용할 수 있는 경우 옵션으로 제공됩니다.

변수	세부 정보
"openshift_logging_storage_kind"	설치 프로그램이 로깅 서비스에 대한 NFS PV를 생성하도록 'NFS'로 설정합니다.
"openshift_logging_storage_host"를 선택합니다	NFS 호스트의 호스트 이름 또는 IP 주소입니다. 가상 머신의 데이터 LIF로 설정해야 합니다.
'openshift_logging_storage_nfs_directory	NFS 내보내기의 마운트 경로입니다. 예를 들어 볼륨이 '/openshift_logging'으로 가정되는 경우 이 변수에 해당 경로를 사용합니다.
'openshift_logging_storage_volume_name'	생성할 PV의 이름(예: PV_ose_logs)입니다.
"openshift_logging_storage_volume_size"	NFS 내보내기의 크기(예: 100Gi)입니다.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 관리자는 동적 프로비저닝을 사용하여

볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
"openshift_logging_es_pvc_dynamic"	동적으로 프로비저닝된 볼륨을 사용하려면 true로 설정합니다.
'openshift_logging_es_pvc_storage_class_name'	PVC에 사용될 스토리지 클래스의 이름입니다.
"openshift_logging_es_pvc_size"를 선택합니다	PVC에서 요청된 체적의 크기입니다.
"openshift_logging_es_pvc_prefix"	로깅 서비스에서 사용하는 PVC의 접두사입니다.
"openshift_logging_es_ops_pvc_dynamic"	작업 로깅 인스턴스에 동적으로 프로비저닝된 볼륨을 사용하려면 "true"로 설정합니다.
'openshift_logging_es_ops_pvc_storage_class_name'	작업 로깅 인스턴스에 대한 스토리지 클래스의 이름입니다.
"openshift_logging_es_ops_pvc_size"를 선택합니다	작업 인스턴스에 대한 볼륨 요청의 크기입니다.
"openshift_logging_es_ops_pvc_prefix"	ops instance PVCs(ops 인스턴스 PVC)의 접두사입니다.

로깅 스택을 배포합니다

초기 OpenShift 설치 프로세스의 일부로 로깅을 배포하는 경우 표준 배포 프로세스만 따르면 됩니다. Ansible이 완료되는 즉시 서비스를 이용할 수 있도록 필요한 서비스와 OpenShift 개체를 구성 및 배포합니다.

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 설명서"를 읽고 따르십시오.

메트릭 서비스

메트릭 서비스는 관리자에게 OpenShift 클러스터의 상태, 리소스 활용도 및 가용성에 대한 중요한 정보를 제공합니다. 또한 POD 자동 확장 기능도 필요하며, 많은 조직에서 비용 청구 및/또는 애플리케이션 표시를 위해 메트릭 서비스의 데이터를 사용합니다.

로깅 서비스 및 OpenShift와 마찬가지로 Ansible을 사용하여 메트릭 서비스를 배포합니다. 또한 로깅 서비스와 마찬가지로 클러스터 초기 설정 중에 또는 구성 요소 설치 방법을 사용하여 작동 후에 메트릭 서비스를 구축할 수 있습니다. 다음 표에는 메트릭 서비스에 대한 영구 스토리지를 구성할 때 중요한 변수가 나와 있습니다.

참고	아래 표에는 메트릭 서비스와 관련된 스토리지 구성과 관련된 변수만 포함되어 있습니다. 문서에 나와 있는 다른 많은 옵션은 배포 내용에 따라 검토, 구성 및 사용해야 합니다.

변수	세부 정보
"openshift_metrics_storage_kind"	설치 프로그램이 로깅 서비스에 대한 NFS PV를 생성하도록 'NFS'로 설정합니다.
'openshift_metrics_storage_host	NFS 호스트의 호스트 이름 또는 IP 주소입니다. SVM의 데이터 LIF로 설정해야 합니다.
'openshift_metrics_storage_nfs_directory	NFS 내보내기의 마운트 경로입니다. 예를 들어, 볼륨이 '/openshift_metrics'로 가정되는 경우 이 변수에 해당 경로를 사용합니다.
'openshift_metrics_storage_volume_name'	생성할 PV의 이름(예: PV_ose_metrics).

변수	세부 정보
'openshift_metrics_storage_volume_size	NFS 내보내기의 크기(예: 100Gi)입니다.

OpenShift 클러스터가 이미 실행 중이고 Trident가 배포 및 구성된 경우 설치 관리자는 동적 프로비저닝을 사용하여 볼륨을 생성할 수 있습니다. 다음 변수를 구성해야 합니다.

변수	세부 정보
'openshift_metrics_cassandra_pvc_prefix'	지표 PVC에 사용할 접두사입니다.
'openshift_metrics_cassandra_pvc_size	요청할 볼륨의 크기입니다.
'openshift_metrics_cassandra_storage_type'	메트릭에 사용할 스토리지 유형으로, 적절한 스토리지 클래스로 PVC를 생성하려면 Ansible에서 이를 동적 으로 설정해야 합니다.
'openshift_metrics_cassanda_pvc_storage_class_name'	사용할 스토리지 클래스의 이름입니다.

메트릭 서비스를 구축합니다

호스트/인벤토리 파일에 정의된 적절한 Ansible 변수를 사용하여 서비스를 구축하십시오. OpenShift 설치 시 배포하는 경우 PV가 자동으로 생성되고 사용됩니다. 구성 요소 플레이북을 사용하여 배포하려는 경우 OpenShift 설치 후 Ansible은 필요한 PVC를 생성하고 Trident에서 이를 위해 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.

Trident 데이터 보호 및 재해 복구

Trident 및 Trident를 사용하여 생성된 볼륨의 보호 및 복구 옵션에 대해 알아보십시오. 지속성 요구사항이 있는 각 애플리케이션에 대한 데이터 보호 및 복구 전략이 있어야 합니다.

Trident 복제 및 복구

재해 발생 시 Trident 복원을 위한 백업을 생성할 수 있습니다.

Trident 복제

Trident는 Kubernetes CRD를 사용하여 자체 상태를 저장 및 관리하고 Kubernetes 클러스터 etcd를 사용하여 메타데이터를 저장합니다.

단계

1. 를 사용하여 Kubernetes 클러스터를 백업합니다 "Kubernetes: etcd 클러스터 백업".
2. FlexVol volume에 백업 아티팩트를 배치합니다

참고	NetApp은 FlexVol이 다른 SVM과 SnapMirror 관계를 맺고 있는 SVM을 보호하는 것이 좋습니다.

Trident 복구

Kubernetes CRD 및 Kubernetes 클러스터 etcd 스냅샷을 사용하여 Trident를 복구할 수 있습니다.

단계

1. 대상 SVM에서 Kubernetes etcd 데이터 파일 및 인증서가 포함된 볼륨을 마스터 노드로 설정할 호스트에 마운트합니다.
2. 에서 Kubernetes 클러스터와 관련된 모든 필수 인증서를 복사합니다 `/etc/kubernetes/pki` 및 에 있는 etcd 멤버 파일 `/var/lib/etcd`.
3. 를 사용하여 etcd 백업에서 Kubernetes 클러스터를 복원합니다 "Kubernetes: etcd 클러스터 복구".
4. 실행 `kubectl get crd` 모든 Trident 사용자 지정 리소스가 준비되었는지 확인하고 Trident 객체를 검색하여 모든 데이터를 사용할 수 있는지 확인합니다.

SVM 복제 및 복구

Trident은 복제 관계를 구성할 수 없지만 스토리지 관리자는 를 사용하여 SVM을 복제할 수 있습니다 "ONTAP SnapMirror를 참조하십시오".

재해가 발생할 경우 SnapMirror 대상 SVM을 활성화하여 데이터 제공을 시작할 수 있습니다. 시스템이 복원되면 운영 시스템으로 다시 전환할 수 있습니다.

이 작업에 대해

SnapMirror SVM 복제 기능을 사용할 때는 다음 사항을 고려하십시오.

- SVM-DR이 활성화된 각 SVM에 대해 별개의 백엔드를 생성해야 합니다.
- SVM-DR을 지원하는 백엔드에 복제를 프로비저닝하지 않아도 되는 볼륨을 가질 필요가 없도록 필요한 경우에만 복제된 백엔드를 선택하도록 스토리지 클래스를 구성합니다.
- 애플리케이션 관리자는 복제와 관련된 추가 비용 및 복잡성을 이해하고 이 프로세스를 시작하기 전에 복구 계획을 신중하게 고려해야 합니다.

SVM 복제

을 사용할 수 있습니다 "ONTAP: SnapMirror SVM 복제" SVM 복제 관계를 생성합니다.

SnapMirror를 사용하여 복제할 항목을 제어하는 옵션을 설정할 수 있습니다. 미리 만들 때 선택한 옵션을 알아야 Trident를 사용한 SVM 복구합니다.

- "-identity -true 를 유지합니다" 전체 SVM 구성을 복제합니다.
- "-discard-configs 네트워크" LIF 및 관련 네트워크 설정은 제외됩니다.
- "-identity -false 를 유지합니다" 볼륨 및 보안 구성만 복제합니다.

Trident를 사용한 SVM 복구

Trident는 SVM 장애를 자동으로 감지하지 못합니다. 재해가 발생할 경우 관리자는 새로운 SVM으로 Trident 페일오버를 수동으로 시작할 수 있습니다.

단계

1. SnapMirror의 예약된 전송 및 진행 중인 전송을 취소하고 복제 관계를 중지한 다음, 소스 SVM을 중지하고 SnapMirror 대상 SVM을 활성화합니다.
2. 를 지정한 경우 `-identity-preserve false` 또는 `-discard-config network` SVM 복제를 구성할 때 를 업데이트합니다 `managementLIF` 및 `dataLIF` Trident 백엔드 정의 파일
3. 확인합니다 `storagePrefix` Trident 백엔드 정의 파일에 있습니다. 이 매개 변수는 변경할 수 없습니다. 생략합니다 `storagePrefix` 백엔드 업데이트가 실패합니다.
4. 다음을 사용하여 새로운 대상 SVM 이름을 반영하도록 필수 백엔드를 업데이트합니다.

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. 를 지정한 경우 `-identity-preserve false` 또는 `discard-config network`모든 애플리케이션 포드를 바운스해야 합니다.

참고	지정한 경우 `-identity-preserve true`, 대상 SVM이 활성화될 때 Trident에서 프로비저닝한 모든 볼륨이 데이터 제공을 시작합니다.

볼륨 복제 및 복구

Trident는 SnapMirror 복제 관계를 구성할 수 없지만, 스토리지 관리자는 를 사용하여 Trident에서 생성된 볼륨을 복제할 수 "ONTAP SnapMirror 복제 및 복구"있습니다.

그런 다음 를 사용하여 복구된 볼륨을 Trident로 가져올 수 "Tridentctl 볼륨 가져오기"있습니다.

참고	볼륨 가져오기는 `ontap-san-economy` 드라이버에서 지원됩니다. `ontap-nas-economy` 드라이버를 사용한 Qtree 가져오기는 기술 미리 보기 기능이며 프로덕션 용도로는 지원되지 않습니다. 가져오기는 `ontap-flexgroup-economy` 드라이버에서 지원되지 않습니다. 자세한 내용은 "Trident 볼륨 가져오기"를 참조하십시오.

스냅샷 데이터 보호

다음을 사용하여 데이터를 보호하고 복원할 수 있습니다.

- PVS(영구 볼륨)의 Kubernetes 볼륨 스냅샷을 생성하는 외부 스냅샷 컨트롤러 및 CRD

  "볼륨 스냅숏"

- ONTAP 스냅샷 - 볼륨의 전체 내용을 복원하거나 개별 파일 또는 LUN을 복구합니다.

  "ONTAP 스냅샷"

Trident 사용하여 상태 저장 애플리케이션의 장애 조치 자동화

Trident의 강제 분리 기능을 사용하면 Kubernetes 클러스터에서 비정상적인 노드의 볼륨을 자동으로 분리하여 데이터 손상을 방지하고 애플리케이션 가용성을 보장할 수 있습니다. 이 기능은 노드가 응답하지 않거나 유지 관리를 위해 오프라인으로 전환되는 시나리오에서 특히

유용합니다.

강제 분리에 대한 세부 정보

강제 분리가 가능합니다. `ontap-san` , `ontap-san-economy` , `ontap-nas` , 그리고 `ontap-nas-economy` 오직. 강제 분리를 활성화하기 전에 Kubernetes 클러스터에서 비정상적 노드 종료(NGNS)를 활성화해야 합니다. NGNS는 Kubernetes 1.28 이상에서 기본적으로 활성화되어 있습니다. 자세한 내용은 다음을 참조하세요. "Kubernetes: 노드 정상 종료 아님" .

참고	또는 `ontap-nas-economy` 드라이버를 사용할 경우 `ontap-nas`, Trident이 관리형 엑스포트 정책을 사용하여 적용된 태그로 인해 Kubernetes 노드에서 액세스를 제한할 수 있도록 백엔드 구성에서 매개변수를 로 `true` 설정해야 `autoExportPolicy` 합니다.
경고	Trident는 Kubernetes ngns를 사용하기 때문에 허용할 수 없는 모든 워크로드의 일정이 재조정될 때까지 비정상 노드에서 테인트를 제거하지 마십시오 `out-of-service`. 무모하게 타트를 적용하거나 제거하면 백엔드 데이터 보호가 위태롭게 될 수 있습니다.

Kubernetes 클러스터 관리자가 노드에 태그를 `enableForceDetach` 적용하고 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 로 설정하면 `true` Trident이 노드 상태와 다음을 확인합니다.

1. 해당 노드에 마운트된 볼륨에 대한 백엔드 I/O 액세스를 중지합니다.
2. Trident 노드 개체를 로 `dirty` 표시합니다(새 발행물에 안전하지 않음).

참고	Trident 컨트롤러는 Trident 노드 포드에 의해 노드가 다시 검증될 때까지(로 표시된 후) 새로운 게시 볼륨 요청을 거부합니다 `dirty`. Trident가 노드를 확인할 수 있을 때까지(새 발행물에 안전함) 마운트된 PVC로 예약된 모든 작업 부하(클러스터 노드가 정상 및 준비 상태임 이후에도)는 수락되지 `clean` 않습니다.

노드 상태가 복원되고 정점이 제거되면 Trident는 다음을 수행합니다.

1. 노드에서 오래된 게시된 경로를 식별하고 제거합니다.
2. 노드가 상태(서비스 중단 시간이 제거되고 노드가 상태)이고 모든 오래되고 `Ready` 게시된 경로가 정리된 경우 `cleanable`, Trident는 노드를 로 재전송하고 게시된 새로운 볼륨을 노드에 허용합니다. `clean`

자동 장애 조치에 대한 세부 정보

통합을 통해 강제 분리 프로세스를 자동화할 수 있습니다."노드 상태 점검(NHC) 운영자" . 노드 장애가 발생하면 NHC는 Trident 노드 복구(TNR)를 트리거하고 Trident의 네임스페이스에 장애가 발생한 노드를 정의하는 TridentNodeRemediation CR을 생성하여 자동으로 강제 분리합니다. TNR은 노드 장애가 발생한 경우에만 생성되며, 노드가 다시 온라인 상태가 되거나 노드가 삭제되면 NHC에서 제거됩니다.

실패한 노드 포드 제거 프로세스

자동 장애 조치는 실패한 노드에서 제거할 작업 부하를 선택합니다. TNR이 생성되면 TNR 컨트롤러는 노드를 더티로 표시하여 새로운 볼륨 게시를 방지하고 강제 분리가 지원되는 포드와 해당 볼륨 첨부 파일을 제거하기 시작합니다.

강제 분리가 지원하는 모든 볼륨/PVC는 자동 장애 조치가 지원됩니다.

- NAS 및 자동 내보내기 정책을 사용하는 NAS 경제 볼륨(SMB는 아직 지원되지 않음).
- SAN 및 SAN 경제 볼륨.

참조하다강제 분리에 대한 세부 정보 .

기본 동작:

- 강제 분리가 지원하는 볼륨을 사용하는 Pod는 실패한 노드에서 제거됩니다. 쿠버네티스는 이를 정상적인 노드로 다시 예약합니다.
- Trident 볼륨이 아닌 볼륨을 포함하여 강제 분리가 지원되지 않는 볼륨을 사용하는 Pod는 실패한 노드에서 제거되지 않습니다.
- 상태 비저장 포드(PVC 아님)는 포드 주석이 없는 한 실패한 노드에서 제거되지 않습니다. `trident.netapp.io/podRemediationPolicy: delete` 설정되었습니다.

포드 제거 동작 재정의:

Pod 제거 동작은 Pod 주석을 사용하여 사용자 정의할 수 있습니다. `trident.netapp.io/podRemediationPolicy[retain, delete]` . 이러한 주석은 장애 조치가 발생할 때 검사되어 사용됩니다. 페일오버 후 주석이 사라지는 것을 방지하려면 Kubernetes 배포/복제 세트 포드 사양에 주석을 적용합니다.

- `retain`- 자동 장애 조치 중에는 실패한 노드에서 Pod가 제거되지 않습니다.
- `delete`- 자동 장애 조치 중에 실패한 노드에서 Pod가 제거됩니다.

이러한 주석은 모든 포드에 적용될 수 있습니다.

경고	• 강제 분리를 지원하는 볼륨의 경우 실패한 노드에서만 I/O 작업이 차단됩니다. • 강제 분리를 지원하지 않는 볼륨의 경우 데이터 손상 및 다중 연결 문제가 발생할 위험이 있습니다.

트라이던트노드복구 **CR**

TridentNodeRemediation(TNR) CR은 실패한 노드를 정의합니다. TNR의 이름은 실패한 노드의 이름입니다.

TNR 예시:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR 상태: 다음 명령을 사용하여 TNR 상태를 확인하세요.
`kubectl get tnr <name> -n <trident-namespace>`

TNR은 다음 상태 중 하나에 속할 수 있습니다.

- 수정:

- 해당 노드에 마운트된 강제 분리가 지원하는 볼륨에 대한 백엔드 I/O 액세스를 중단합니다.
- Trident 노드 객체가 더티(새로운 게시물에 안전하지 않음)로 표시되었습니다.
- 노드에서 포드 및 볼륨 첨부 파일 제거

- 노드 복구 보류:
  - 컨트롤러는 노드가 다시 온라인 상태가 될 때까지 기다리고 있습니다.
  - 노드가 온라인 상태가 되면 publish-enforcement를 통해 노드가 정리되고 새로운 볼륨 출판에 적합한 상태가 됩니다.
- 노드가 K8s에서 삭제되면 TNR 컨트롤러는 TNR을 제거하고 조정을 중단합니다.
- 성공:
  - 모든 수정 및 노드 복구 단계가 성공적으로 완료되었습니다. 노드가 정리되어 새로운 권의 출판에 적합합니다.
- 실패한:
  - 복구할 수 없는 오류입니다. 오류 이유는 CR의 status.message 필드에 설정됩니다.

자동 장애 조치 활성화

필수 조건:

- 자동 장애 조치를 활성화하기 전에 강제 분리가 활성화되어 있는지 확인하세요. 자세한 내용은 다음을 참조하세요 .강제 분리에 대한 세부 정보 .
- Kubernetes 클러스터에 노드 상태 점검(NHC)을 설치합니다.
  - "operator-sdk 설치".
  - 클러스터에 Operator Lifecycle Manager(OLM)가 아직 설치되지 않았다면 설치하세요. `operator-sdk olm install` .
  - 노드 상태 점검 운영자 설치: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` .

> 참고 아래 사용자 정의 노드 상태 점검 솔루션 통합 섹션에 명시된 대로 노드 장애를 감지하는 다른 방법을 사용할 수도 있습니다.

보다"노드 상태 점검 운영자" 자세한 내용은.

단계

1. 클러스터의 워커 노드를 모니터링하기 위해 Trident 네임스페이스에 NodeHealthCheck(NHC) CR을 만듭니다. 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. 노드 상태 점검 CR을 적용합니다. `trident` 네임스페이스.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

위 CR은 노드 조건 '준비: 거짓' 및 '알 수 없음'에 대해 K8s 워커 노드를 감시하도록 구성되어 있습니다. 자동 장애 조치는 노드가 준비: 거짓 또는 준비: 알 수 없음 상태로 전환되면 실행됩니다.

그만큼 `unhealthyConditions` CR에서는 0초의 유예 기간을 사용합니다. 이렇게 하면 K8s가 노드 조건 Ready: false를 설정하는 즉시 자동 장애 조치가 트리거됩니다. 이 조건은 K8s가 노드에서 하트비트를 잃은 후에 설정됩니다. K8s는 마지막 하트비트 이후 Ready: false로 설정되기 전에 기본적으로 40초간 기다립니다. 이 유예 기간은 K8s 배포 옵션에서 사용자 정의할 수 있습니다.

추가 구성 옵션은 다음을 참조하세요."Node-Healthcheck-Operator 문서" .

추가 설정 정보

Trident 강제 분리가 활성화된 상태로 설치하면 NHC와의 통합을 용이하게 하기 위해 Trident 네임스페이스에 두 개의 추가 리소스인 TridentNodeRemediationTemplate(TNRT)과 ClusterRole이 자동으로 생성됩니다.

TridentNodeRemediationTemplate(TNRT):

TNRT는 NHC 컨트롤러를 위한 템플릿 역할을 하며, NHC 컨트롤러는 필요에 따라 TNRT를 사용하여 TNR 리소스를 생성합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
```

```
    spec: {}
```

클러스터 역할:

강제 분리가 활성화된 경우 설치 중에 클러스터 역할도 추가됩니다. 이를 통해 NHC는 Trident 네임스페이스의 TNR에 대한 권한을 얻습니다.

```
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 중지합니다. 위에 설명된 NHC CR을 일시 중지하려면 CR을 패치하면 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 제거하세요.

제한 사항

- 강제 분리가 지원되는 볼륨의 경우 실패한 노드에서만 I/O 작업이 방지됩니다. 강제 분리를 지원하는 볼륨/PVC를 사용하는 포드만 자동으로 제거됩니다.
- 자동 장애 조치 및 강제 분리 기능은 트라이던트 컨트롤러 포드 내부에서 실행됩니다. 트라이던트 컨트롤러를 호스팅하는 노드에 장애가 발생하면 K8s가 포드를 정상적인 노드로 옮길 때까지 자동 장애 조치가 지연됩니다.

사용자 정의 노드 상태 점검 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 솔루션이 다음을 수행해야 합니다.

- 노드 장애가 감지되면 장애가 발생한 노드의 이름을 TNR CR 이름으로 사용하여 TNR을 생성합니다.
- 노드가 복구되고 TNR이 성공 상태가 되면 TNR을 삭제합니다.

보안

Trident 보안 권장 사항

여기에 나열된 권장 사항을 사용하여 Trident가 안전하게 설치되었는지 확인하십시오.

자체 네임스페이스에서 **Trident** 실행

안정적인 스토리지를 확보하고 잠재적 악의적인 활동을 차단하려면 애플리케이션, 애플리케이션 관리자, 사용자 및 관리 애플리케이션이 Trident 오브젝트 정의 또는 포드에 액세스하지 못하도록 방지하는 것이 중요합니다.

Trident에서 다른 애플리케이션과 사용자를 분리하려면 항상 자체 Kubernetes 네임스페이스에 Trident를 (`trident`설치해야 합니다.) Trident를 자체 네임스페이스에 배치하면 Kubernetes 관리 담당자만 Trident Pod와 이름 속도의 CRD 개체에 저장된 아티팩트(예: 백엔드 및 CHAP 암호(해당하는 경우)에 액세스할 수 있습니다. 관리자만 Trident 네임스페이스에 액세스하여 응용 프로그램에 액세스할 수 있도록 해야 `tridentctl 합니다.

ONTAP SAN 백엔드에 **CHAP** 인증을 사용합니다

Trident는 및 ontap-san-economy 드라이버를 사용하여 ONTAP SAN 워크로드에 대한 CHAP 기반 인증을 ontap-san 지원합니다. NetApp에서는 Trident 호스트와 스토리지 백엔드 간의 인증에 양방향 CHAP를 사용할 것을 권장합니다.

SAN 스토리지 드라이버를 사용하는 ONTAP 백엔드의 경우 Trident에서 양방향 CHAP를 설정하고 를 통해 CHAP 사용자 이름 및 암호를 관리할 수 tridentctl 있습니다. Trident가 ONTAP 백엔드에서 CHAP를 구성하는 방법에 대한 자세한 내용은 을 "ONTAP SAN 드라이버를 사용하여 백엔드를 구성할 준비를 합니다"참조하십시오.

NetApp HCI 및 **SolidFire** 백엔드에서 **CHAP** 인증을 사용합니다

양방향 CHAP를 구축하여 호스트와 NetApp HCI 및 SolidFire 백엔드 간의 인증을 보장하는 것이 좋습니다. Trident는 테넌트당 두 개의 CHAP 암호를 포함하는 비밀 개체를 사용합니다. Trident가 설치되면 CHAP 암호를 관리하고 tridentvolume 해당 PV에 대한 CR 개체에 저장합니다. PV를 생성할 때 Trident는 CHAP 암호를 사용하여 iSCSI 세션을 시작하고 CHAP를 통해 NetApp HCI 및 SolidFire 시스템과 통신합니다.

참고	Trident에 의해 생성된 볼륨은 볼륨 액세스 그룹과 연결되지 않습니다.

NVE 및 **NAE**에 **Trident**를 사용하십시오

NetApp ONTAP는 유휴 데이터 암호화를 제공하여 디스크를 도난, 반환 또는 용도 변경할 때 중요한 데이터를 보호합니다. 자세한 내용은 을 참조하십시오 "NetApp 볼륨 암호화 구성 개요".

- 백엔드에서 NAE가 활성화된 경우 Trident에서 프로비저닝된 모든 볼륨은 NAE가 사용됩니다.

◦ NVE 암호화 플래그를 로 설정하여 NAE 지원 볼륨을 생성할 수 있습니다 `""`.

- 백엔드에서 NAE가 사용되지 않는 경우, 백엔드 구성에서 NVE 암호화 플래그가 (기본값)로 설정되지 않은 경우 Trident에서 프로비저닝된 모든 볼륨은 NVE가 `false` 사용됩니다.

참고

NAE 지원 백엔드의 Trident에서 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.
- NVE 암호화 플래그를 NAE 지원 백엔드에서 ON으로 `false` 설정하면 NAE 지원 볼륨이 생성됩니다. NVE 암호화 플래그를 로 설정하여 NAE 암호화를 비활성화할 수 `false` 없습니다.

- Trident에서 NVE 암호화 플래그를 로 명시적으로 설정하여 NVE 볼륨을 수동으로 생성할 수 `true` 있습니다.

백엔드 구성 옵션에 대한 자세한 내용은 다음을 참조하십시오.

- "ONTAP SAN 구성 옵션"
- "ONTAP NAS 구성 옵션"

Trident Linux Unified Key Setup(LUKS)

Linux Unified Key Setup(LUKS)은 디스크 암호화 사양입니다. Trident는 LUKS를 사용하여 ONTAP SAN 및 ONTAP SAN ECONOMY 볼륨에 대해 볼륨별 호스트 측 암호화를 제공하며, 암호 순환 및 볼륨 확장을 지원합니다.

Trident에서 LUKS로 암호화된 볼륨은 에서 권장하는 AES-XTS-plar64 사이퍼 및 모드를 "NIST"사용합니다.

참고

ASA r2 시스템에서는 LUKS 암호화가 지원되지 않습니다. ASA r2 시스템에 대한 자세한 내용은 다음을 참조하세요. "ASA R2 스토리지 시스템에 대해 알아보십시오" .

시작하기 전에

- 작업자 노드에는 Cryptsetup 2.1 이상(3.0 이하)이 설치되어 있어야 합니다. 자세한 내용은 를 참조하십시오 "Gitlab: cryptsetup".
- 성능상의 이유로 NetApp에서는 작업자 노드가 AES-NI(Advanced Encryption Standard New Instructions)를 지원하는 것이 좋습니다. AES-NI 지원을 확인하려면 다음 명령을 실행합니다.

```
grep "aes" /proc/cpuinfo
```

아무 것도 반환되지 않으면 프로세서는 AES-NI를 지원하지 않습니다. AES-NI에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오. "인텔: AES-NI(Advanced Encryption Standard Instructions)".

LUKS 암호화를 사용합니다

ONTAP SAN 및 ONTAP SAN 이코노미 볼륨에 대해 Linux 통합 키 설정(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 을 참조하십시오 "ONTAP SAN 구성 옵션".

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. 사용 `parameters.selector` LUKS 암호화를 사용하여 스토리지 풀을 정의합니다. 예를 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 암호를 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
```

RWX 원시 블록 볼륨과 함께 LUKS 사용

Trident 26.10부터는 `ReadWriteMany`(RWX) 액세스 모드를 사용하여 여러 Kubernetes 노드에 연결된 iSCSI 기반 원시 블록 볼륨에 LUKS 암호화를 사용할 수 있습니다. 이는 라이브 마이그레이션 중에 소스 노드에 연결된 상태에서 대상 노드에 동일한 암호화된 볼륨을 연결해야 하는 KubeVirt 가상 머신 등 워크로드에 유용합니다.

RWX LUKS 지원은 원시 블록 볼륨으로 제한됩니다. 파일 시스템 모드 RWX LUKS 볼륨은 지원되지 않습니다. 애플리케이션은 복호화된 데이터에 대한 동시 접근을 조정할 책임이 있으며, Trident는 LUKS 작업만 조정합니다.

LUKS 암호화를 사용합니다에 설명된 대로 백엔드, StorageClass 및 암호 Secret에 LUKS를 구성합니다. 그런 다음 `volumeMode: Block` 및 `ReadWriteMany`을(를) 사용하여 PVC를 생성합니다:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-rwx-pvc
  namespace: my-app
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  storageClassName: san-luks-rwx
  resources:
    requests:
      storage: 10Gi
```

여러 노드의 Pod는 이 PVC를 원시 블록 장치로 연결할 수 있습니다. Trident는 처음 연결 시 LUKS2로 장치를 포맷하고 이후 노드에서 기존에 암호화된 볼륨을 엽니다. 암호 변경 및 볼륨 확장 등 기존 LUKS 작업이 지원됩니다.

Trident가 LUKS 연산을 직렬화하는 이유

LUKS2는 암호화 헤더의 중복 복사본을 블록 장치 시작 부분에 저장합니다. `cryptsetup`은(는) 복사본이 서로 다름을 감지할 때, 읽기 전용으로 보이는 명령을 실행하는 동안을 포함하여 헤더를 복구할 수 있습니다. 잠금은 노드에 국한되며 다른 노드의 작업을 조정할 수 없습니다. 한 노드가 헤더를 업데이트하는 동안 다른 노드가 `cryptsetup`을(를) 실행하면, 두 번째 노드가 헤더 복구를 트리거하여 업데이트를 중단시키고 헤더를 손상시킬 수 있습니다.

동시 헤더 접근을 방지하기 위해 Trident는 각 RWX LUKS 볼륨에 대해 Kubernetes 리스를 사용합니다. 노드가 볼륨에 대한 `cryptsetup` 명령을 실행하기 전에 리스를 확보해야 합니다. Trident는 작업이 성공적으로 완료되면 리스를 해제합니다. 리스가 확보된 동안 다른 노드가 작업을 요청하면 Trident는 재시도 가능한 오류를 반환하고 Kubernetes는 나중에 재시도합니다. 서로 다른 볼륨에 대한 작업은 독립적입니다.

`cryptsetup` 작업이 실패하면 Trident는 의도적으로 리스를 유지하여 동일한 노드에서만

재시도할 수 있도록 합니다. 이유가 `LeaseRetained`인 경고 이벤트에 실패가 기록됩니다. 리스는 자동으로 만료되지 않습니다. 관리자는 해당 보유 노드에서 `cryptsetup` 작업이 실행되고 있지 않음을 확인한 후에만 리스를 제거해야 합니다. 복구 단계는 xref:{relative_path}rwx-luks-stuck-lease.html["RWX Linux Unified Key Setup(LUKS) 볼륨에 대한 임대가 중단된 문제를 해결합니다"]을(를) 참조하십시오.

노드들이 순차적으로 볼륨을 열기 때문에, 여러 노드에서 워크로드를 시작하는 데 걸리는 시간이 암호화되지 않은 볼륨을 사용할 때보다 오래 걸릴 수 있습니다. Kubernetes는 현재 작업이 완료된 후 일시적인 잠금 오류를 재시도합니다. 대규모 스케일아웃 시 Pod는 `ContainerCreating`에 더 오래 머무를 수 있으며, 동일한 노드의 워크로드는 단일 열린 장치 매핑을 공유할 수 있습니다.

제한 사항

LUKS - 암호화된 볼륨은 ONTAP 중복 제거 및 압축을 활용할 수 없습니다.

LUKS 볼륨을 가져오기 위한 백엔드 구성입니다

LUKS 볼륨을 가져오려면 백엔드에서 로(`true` 설정해야 `luksEncryption` 합니다. 이 `luksEncryption` 옵션은 볼륨이 LUKS 호환(`true`) 또는 LUKS 규격이 아닌 경우 Trident에 지시합니다.(`false`

```
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 볼륨을 가져오기 위한 **PVC** 구성입니다

LUKS 볼륨을 동적으로 가져오려면 `trident.netapp.io/luksEncryption` `true` 이 예제에 표시된 대로 주석을 로 설정하고 PVC에 LUKS 사용 스토리지 클래스를 포함합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
```

```
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

LUKS 암호를 회전합니다

LUKS 암호를 회전하고 회전을 확인할 수 있습니다.

경고	볼륨, 스냅샷 또는 비밀이 더 이상 참조하지 않음을 확인할 때까지 암호문을 잊지 마십시오. 참조된 암호가 손실된 경우 볼륨을 마운트할 수 없으며 데이터가 암호화된 상태로 유지되고 액세스할 수 없게 됩니다.

이 작업에 대해

LUKS 암호 회전은 새 LUKS 암호를 지정한 후 볼륨을 마운트하는 POD가 생성될 때 발생합니다. 새 Pod가 생성되면 Trident는 볼륨의 LUKS 암호를 암호의 활성 암호와 비교합니다.

- 볼륨의 암호가 비밀의 활성 암호와 일치하지 않으면 회전이 발생합니다.
- 볼륨의 암호가 비밀의 활성 암호와 일치하면 가 됩니다 `previous-luks-passphrase` 매개 변수는 무시됩니다.

단계

1. 를 추가합니다 `node-publish-secret-name` 및 `node-publish-secret-namespace` StorageClass 매개 변수입니다. 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 스냅숏에서 기존 암호를 식별합니다.

   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. 볼륨에 대한 LUKS 암호를 업데이트하여 새 암호 및 이전 암호 문구를 지정합니다. 확인합니다 `previous-luke-passphrase-name` 및 `previous-luks-passphrase` 이전 패스프레이즈를 일치시킵니다.

```
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 포드를 생성합니다. 이 작업은 회전을 시작하는 데 필요합니다.
5. 패스프레이즈가 회전되었는지 확인합니다.

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

볼륨과 스냅샷에 새 암호문만 반환되면 암호가 회전되었습니다.

참고 예를 들어, 두 개의 암호 구문이 반환되는 경우 `luksPassphraseNames: ["B", "A"]`, 회전이

> 완료되지 않았습니다. 새 포드를 트리거하여 회전을 완료할 수 있습니다.

볼륨 확장을 설정합니다

LUKS 암호화 볼륨에서 볼륨 확장을 활성화할 수 있습니다.

단계

1. 를 활성화합니다 `CSINodeExpandSecret` 기능 게이트(베타 1.25+). 을 참조하십시오 "Kubernetes 1.25: CSI 볼륨의 노드 기반 확장에 비밀을 사용합니다" 를 참조하십시오.
2. 를 추가합니다 `node-expand-secret-name` 및 `node-expand-secret-namespace` StorageClass 매개 변수입니다. 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 저장소 확장을 시작할 때 kubelet은 적절한 자격 증명을 드라이버에 전달합니다.

RWX Linux Unified Key Setup(LUKS) 볼륨에 대한 임대가 중단된 문제를 해결합니다

Kubernetes 임대를 보유한 노드에 장애가 발생하면 ReadWriteMany(RWX) Linux 통합 키 설정(LUKS) 볼륨이 차단될 수 있습니다. 임대를 삭제하기 전에 실행 중인 `cryptsetup` 작업이 없는지 확인하십시오. 작업 중에 임대를 삭제하면 LUKS 헤더가 손상될 위험이 있습니다.

임대 이해

Trident는 Kubernetes 리스를 사용하여 한 번에 하나의 노드만 RWX LUKS 볼륨에서 `cryptsetup` 명령을 실행하도록 합니다. 리스는 Trident 네임스페이스에 생성되며, 리스 이름은 PersistentVolume(PV) 이름입니다. 홀더 ID는 작업을 실행하는 노드를 식별합니다.

Trident는 작업이 성공적으로 완료되면 리스를 삭제합니다. 작업이 실패하면 Trident는 동일한 노드에서만 재시도할 수 있도록 리스를 유지합니다. 리스는 자동으로 만료되거나 갱신되지 않습니다. 이유가 `LeaseRetained`인 경고 이벤트에는 실패한 작업에 대한 세부 정보가 포함됩니다.

잠재적으로 중단된 **Lease** 식별

증상으로는 포드가 `ContainerCreating` 또는 `Terminating`에 멈춰 있거나, 볼륨 확장이 실패하거나, 일시적인 잠금 오류가 발생하는 경우가 있습니다.

1. 문제가 발생한 포드에서 사용된 PVC를 찾으십시오:

```
kubectl get pod -n <app-namespace> <pod-name> \
  -o jsonpath='{.spec.volumes[*].persistentVolumeClaim.claimName}'
```

2. PVC에 바인딩된 PV를 찾으십시오:

```
kubectl get pvc -n <app-namespace> <pvc-name> \
  -o jsonpath='{.spec.volumeName}'
```

3. Trident 네임스페이스에서 리스를 나열하고 PV 이름으로 리스를 식별합니다:

```
kubectl get lease -n <trident-namespace>
```

4. holder, age, 이벤트를 포함하여 일치하는 Lease를 검사합니다:

```
kubectl describe lease -n <trident-namespace> <pv-name>
```

`cryptsetup` 작업은 일반적으로 몇 초 내에 완료됩니다. 5~10분 이상 된 Lease는 교착 상태를 나타낼 수 있습니다. `LeaseRetained` 이벤트에서 반복되는 오류를 확인하고, 홀더 노드를 식별하고, 노드 상태를 확인하십시오:

```
kubectl get nodes
kubectl get pods -A -o wide --field-selector spec.nodeName=<lease-holder-
node>
```

노드가 정상이고 리스에 경고 이벤트가 없는 경우, 리스가 멈췄다고 판단하기 전에 홀더 노드의 Trident 노드 파드 로그를 확인하십시오:

```
kubectl get pods -n <trident-namespace> -o wide \
  --field-selector spec.nodeName=<lease-holder-node>
kubectl logs -n <trident-namespace> <trident-node-pod> -c trident-main
```

기본 노드 또는 네트워크 문제를 해결할 수 있다면 먼저 해결하십시오. Trident가 작업을 성공적으로 완료하면 리스가 해결될 수 있습니다.

고착된 임대를 안전하게 해결하십시오

중요함	리스 소유자가 여전히 `cryptsetup` 작업을 실행 중일 수 있는 동안에는 리스를 삭제하지 마십시오. 리스로 조정되지 않는 서로 다른 노드의 동시 `cryptsetup` 명령은 LUKS 헤더를 손상시킬 수 있습니다.

노드 상태에 맞는 해결 방법을 사용하십시오:

- 홀더 노드에 장애가 발생하여 전원이 꺼졌습니다**:** 해당 노드가 보유한 리스를 식별하고 영향을 받는 리스를 삭제합니다. 워크로드가 재스케줄링되기 전에 노드가 복구되면 Trident가 작업을 완료하고 리스를 해제하도록 허용합니다. 워크로드가 이미 이동된 경우, 복구된 노드의 Pod가 완전히 종료될 때까지 기다리고, 교착 상태가 여전히 존재하는지 확인한 후 만료된 리스를 제거합니다.
- 홀더 노드의 컨트롤 플레인 연결이 끊어졌습니다**:** 노드에서 여전히 워크로드가 실행 중일 수 있습니다. 리스를 삭제하기 전에 노드를 종료하는 것이 좋습니다. 노드에 계속 액세스할 수 있는 경우, 해당 노드의 Trident 노드 컨테이너를 중지하거나 로그를 통해 리스를 획득하거나 삭제할 수 없고 실행 중이 아님을 확인 `cryptsetup`하십시오. Trident는 각 작업 전에 리스를 획득하기 위해 Kubernetes API에 액세스해야 합니다. 연결이 끊어지면 다른 작업을 시작할 수 없습니다. 진행 중인 명령이 완료될 수 있도록 리스 생성 후 최소 5분 이상 기다린 후 삭제를 고려하십시오.
- 홀더 노드와 네트워크가 정상인 경우**:** Lease 경고 이벤트와 Trident 노드 로그를 검사하여 실패한 작업을 식별하십시오. 영향을 받는 워크로드를 다른 노드로 이동해야 하는 경우, 홀더 노드를 격리하고, 영향을 받는 애플리케이션 Pod를 강제 삭제하지 않고 삭제한 후, Pod가 종료될 때까지 기다린 다음 노드의 격리를 해제하십시오. 원래 작업이 더 이상 실행되지 않는 것을 확인한 후에만 Lease를 삭제하십시오.

안전한지 확인한 후에만 임대를 삭제하십시오:

```
kubectl delete lease -n <trident-namespace> <pv-name>
```

복구 조치 후 영향을 받은 Pod 및 볼륨 작업을 모니터링하십시오. Lease가 다시 유지되는 경우 Lease 이벤트에 보고된 cryptsetup 오류를 조사하고 근본적인 문제를 해결하십시오.

Trident Kerberos 전송 중 암호화

Kerberos 전송 중 암호화를 사용하면 관리 클러스터와 스토리지 백엔드 사이의 트래픽에 암호화를 활성화하여 데이터 액세스 보안을 강화할 수 있습니다.

Trident는 ONTAP를 스토리지 백엔드로 위해 Kerberos 암호화를 지원합니다.

- 온프레미스 **ONTAP** - Trident는 Red Hat OpenShift 및 업스트림 Kubernetes 클러스터에서 온프레미스 ONTAP 볼륨으로의 NFSv3 및 NFSv4 연결을 통해 Kerberos 암호화를 지원합니다.

따라서 스냅샷을 생성하고, 삭제하고, 크기 조정하고, 스냅샷, 클론 읽기 전용 클론 생성 및 NFS 암호화를 사용하는 볼륨 가져오기

사내 **ONTAP** 볼륨을 사용하여 전송 중인 **Kerberos** 암호화를 구성합니다

관리 클러스터와 온프레미스 ONTAP 스토리지 백엔드 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

참고	온-프레미스 ONTAP 스토리지 백엔드를 사용하는 NFS 트래픽에 대한 Kerberos 암호화는 스토리지

> 드라이버를 사용하는 경우에만 `ontap-nas` 지원됩니다.

시작하기 전에

- 유틸리티에 액세스할 수 있는지 `tridentctl` 확인합니다.
- ONTAP 스토리지 백엔드에 대한 관리자 액세스 권한이 있어야 합니다.
- ONTAP 스토리지 백엔드에서 공유할 볼륨의 이름을 알고 있어야 합니다.
- NFS 볼륨에 대한 Kerberos 암호화를 지원하는 ONTAP 스토리지 VM을 준비했는지 확인합니다. 자세한 내용은 을 "데이터 LIF에서 Kerberos 활성화" 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 엑스포트 정책을 추가 또는 수정합니다

기존 ONTAP 엑스포트 정책에 규칙을 추가하거나, ONTAP 스토리지 VM 루트 볼륨뿐 아니라 업스트림 Kubernetes 클러스터와 공유된 ONTAP 볼륨에 대해 Kerberos 암호화를 지원하는 새 엑스포트 정책을 생성해야 합니다. 추가한 내보내기 정책 규칙 또는 새로 만든 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 필요에 따라 세 가지 Kerberos 암호화 버전 중 하나를 구성할 수 있습니다.

- * Kerberos 5 * - (인증 및 암호화)
- * Kerberos 5i * - (인증 및 ID 보호 암호화)
- * Kerberos 5p * - (인증 및 암호화, ID 및 개인 정보 보호)

적절한 액세스 권한을 사용하여 ONTAP 엑스포트 정책 규칙을 구성합니다. 예를 들어, 클러스터가 Kerberos 5i 및 Kerberos 5p 암호화가 혼합된 NFS 볼륨을 마운트하는 경우 다음 액세스 설정을 사용합니다.

유형	읽기 전용 액세스	읽기/쓰기 권한	고급 사용자 액세스
Unix	활성화됨	활성화됨	활성화됨
Kerberos 5i	활성화됨	활성화됨	활성화됨
Kerberos 5p	활성화됨	활성화됨	활성화됨

ONTAP 엑스포트 정책과 엑스포트 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 만들 수 있습니다.

이 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 매개 변수를 사용하여 세 가지 Kerberos 암호화 버전 중 하나를 지정할 수 `spec.nfsMountOptions` 있습니다.

- `spec.nfsMountOptions: sec=krb5` (인증 및 암호화)
- `spec.nfsMountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `spec.nfsMountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다.

단계

1. 관리되는 클러스터에서 다음 예제를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

```
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 생성합니다.

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

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

```
tridentctl logs
```

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

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

이 작업에 대해

저장소 클래스 개체를 만들 때 매개 변수를 사용하여 Kerberos 암호화의 세 가지 버전 중 하나를 지정할 수 `mountOptions` 있습니다.

- `mountOptions: sec=krb5` (인증 및 암호화)
- `mountOptions: sec=krb5i` (인증 및 암호화, ID 보호)
- `mountOptions: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

Kerberos 수준을 하나만 지정하십시오. 매개 변수 목록에서 Kerberos 암호화 수준을 두 개 이상 지정하면 첫 번째 옵션만 사용됩니다. 스토리지 백엔드 구성에서 지정한 암호화 수준이 스토리지 클래스 객체에 지정한 레벨과 다른 경우 스토리지 클래스 객체가 우선합니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 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 확인합니다.

```
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 후 이제 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 을 "볼륨을 프로비저닝합니다"참조하십시오.

Azure NetApp Files 볼륨과 함께 전송 중인 Kerberos 암호화를 구성합니다

관리 클러스터와 단일 Azure NetApp Files 스토리지 백엔드 또는 Azure NetApp Files 스토리지 백엔드의 가상 풀 사이의 스토리지 트래픽에 Kerberos 암호화를 활성화할 수 있습니다.

시작하기 전에

- 관리되는 Red Hat OpenShift 클러스터에서 Trident를 활성화했는지 확인합니다.
- 유틸리티에 액세스할 수 있는지 `tridentctl` 확인합니다.
- 요구 사항을 확인하고 의 지침에 따라 Kerberos 암호화용 Azure NetApp Files 스토리지 백엔드를 준비했는지 "Azure NetApp Files 설명서"확인합니다.
- 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 작업에 대해

Kerberos 암호화를 구성하는 스토리지 백엔드 구성 파일을 만들 때 다음 두 가지 가능한 수준 중 하나에 적용되도록 정의할 수 있습니다.

- 필드를 사용하는 * 스토리지 백엔드 레벨 * `spec.kerberos`
- 필드를 사용하는 * 가상 풀 레벨 * `spec.storage.kerberos`

가상 풀 레벨에서 구성을 정의하면 스토리지 클래스의 레이블을 사용하여 풀이 선택됩니다.

두 레벨에서 Kerberos 암호화의 세 가지 버전 중 하나를 지정할 수 있습니다.

- `kerberos: sec=krb5` (인증 및 암호화)
- `kerberos: sec=krb5i` (인증 및 암호화, ID 보호)
- `kerberos: sec=krb5p` (인증 및 암호화, ID 및 개인 정보 보호)

단계

1. 관리되는 클러스터에서 스토리지 백엔드(스토리지 백엔드 레벨 또는 가상 풀 레벨)를 정의해야 하는 위치에 따라 다음 예제 중 하나를 사용하여 스토리지 백엔드 구성 파일을 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

스토리지 백엔드 레벨의 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 레벨 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 생성합니다.

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

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

```
tridentctl logs
```

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

스토리지 클래스를 생성합니다

스토리지 클래스를 만들어 Kerberos 암호화를 사용하여 볼륨을 프로비저닝할 수 있습니다.

단계

1. 다음 예제를 사용하여 StorageClass Kubernetes 개체를 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 확인합니다.

```
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 후 이제 볼륨을 프로비저닝할 수 있습니다. 자세한 내용은 을 "볼륨을 프로비저닝합니다"참조하십시오.

Trident Protect로 애플리케이션을 보호하세요

Trident Protect에 대해 알아보세요

NetApp Trident Protect는 NetApp ONTAP 스토리지 시스템과 NetApp Trident CSI 스토리지 프로비저너가 지원하는 상태 저장 Kubernetes 애플리케이션의 기능과 가용성을 향상시키는 고급 애플리케이션 데이터 관리 기능을 제공합니다. Trident Protect는 퍼블릭 클라우드와 온프레미스 환경에서 컨테이너화된 워크로드의 관리, 보호 및 이동을 간소화합니다. 또한 API와 CLI를 통해 자동화 기능도 제공합니다.

Trident Protect를 사용하면 사용자 정의 리소스(CR)를 생성하거나 Trident Protect CLI를 사용하여 애플리케이션을 보호할 수 있습니다.

다음 단계

Trident Protect를 설치하기 전에 요구 사항에 대해 알아볼 수 있습니다.

- "Trident 프로텍트 요구 사항"

Trident Protect 설치

Trident 프로텍트 요구 사항

운영 환경, 애플리케이션 클러스터, 애플리케이션 및 라이선스의 준비 상태를 확인하여 시작하세요. Trident Protect를 배포하고 운영하려면 환경이 이러한 요구 사항을 충족하는지 확인하세요.

Trident Protect Kubernetes 클러스터 호환성

Trident Protect는 다음을 포함한 광범위한 완전 관리형 및 자체 관리형 Kubernetes 제품과 호환됩니다.

- Amazon Elastic Kubernetes Service(EKS)
- Google Kubernetes Engine(GKE)
- Microsoft Azure Kubernetes Service(AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0(ONTAP iSCSI)
- 수세 목장
- VMware Tanzu 포트폴리오
- 업스트림 Kubernetes

참고

- Trident Protect 백업은 Linux 컴퓨트 노드에서만 지원됩니다. Windows 컴퓨팅 노드에서는 백업 작업이 지원되지 않습니다.
- Trident Protect를 설치하는 클러스터가 실행 중인 스냅샷 컨트롤러와 관련 CRD로 구성되어 있는지 확인하세요. 스냅샷 컨트롤러를 설치하려면 다음을 참조하세요. "참조하십시오" .

- 하나 이상의 VolumeSnapshotClass가 존재하는지 확인하십시오. 자세한 내용은 "VolumeSnapshotClass"을(를) 참조하십시오.
- Trident Protect를 설치하려면 Helm 4.x 이상이 필요합니다.

Trident Protect 스토리지 백엔드 호환성

Trident Protect는 다음과 같은 스토리지 백엔드를 지원합니다.

- NetApp ONTAP용 Amazon FSx
- Cloud Volumes ONTAP
- ONTAP 스토리지 어레이
- Google Cloud NetApp 볼륨
- Azure NetApp Files

스토리지 백엔드가 다음 요구 사항을 충족하는지 확인합니다.

- 클러스터에 연결된 NetApp 스토리지가 Trident 24.02 이상(Trident 24.10 권장)을 사용하는지 확인하세요.
- NetApp ONTAP 스토리지 백엔드가 있는지 확인합니다.
- 백업을 저장할 오브젝트 스토리지 버킷을 구성했는지 확인합니다.
- 애플리케이션이나 애플리케이션 데이터 관리 작업에 사용할 애플리케이션 네임스페이스를 만듭니다. Trident Protect는 이러한 네임스페이스를 자동으로 생성하지 않습니다. 사용자 정의 리소스에 존재하지 않는 네임스페이스를 지정하면 작업이 실패합니다.

NAS 경제 볼륨에 대한 요구 사항

Trident Protect는 NAS 경제형 볼륨에 대한 백업 및 복원 작업을 지원합니다. 현재 NAS 경제 볼륨에 대한 스냅샷, 복제 및 SnapMirror 복제는 지원되지 않습니다. Trident Protect와 함께 사용하려는 각 NAS-Economy 볼륨에 대해 스냅샷 디렉토리를 활성화해야 합니다.

참고

일부 애플리케이션은 스냅샷 디렉토리를 사용하는 볼륨과 호환되지 않습니다. 이러한 애플리케이션의 경우 ONTAP 스토리지 시스템에서 다음 명령을 실행하여 스냅샷 디렉토리를 숨겨야 합니다.

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

각 NAS 경제 볼륨에 대해 다음 명령을 실행하여 스냅샷 디렉토리를 설정할 수 있으며, 변경할 볼륨의 UUID로 바꿀 수 `<volume-UUID>` 있습니다.

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

참고

Trident 백엔드 구성 옵션을 로 `true` 설정하면 기본적으로 새 볼륨에 대해 스냅샷 디렉토리를 설정할 수 `snapshotDir` 있습니다. 기존 볼륨은 영향을 받지 않습니다.

KubeVirt VM으로 데이터 보호

Trident Protect는 데이터 보호 작업 중에 KubeVirt 가상 머신에 파일 시스템 동결 및 동결 해제 기능을 제공하여 데이터 일관성을 보장합니다. VM 동결 작업에 대한 구성 방법과 기본 동작은 Trident Protect 버전마다 다르며, 최신 릴리스에서는 Helm 차트 매개변수를 통해 간소화된 구성을 제공합니다.

참고	복원 작업 중에 `VirtualMachineSnapshots` 가상 머신(VM)에 대해 생성된 항목은 복원되지 않습니다.

▼ **Trident Protect 25.10** 이상

Trident Protect는 일관성을 보장하기 위해 데이터 보호 작업 중에 KubeVirt 파일 시스템을 자동으로 동결 및 동결 해제합니다. Trident Protect 25.10부터 다음을 사용하여 이 동작을 비활성화할 수 있습니다. `vm.freeze` Helm 차트 설치 중 매개변수. 해당 매개변수는 기본적으로 활성화되어 있습니다.

```
helm install ... --set vm.freeze=false ...
```

▼ **Trident** 프로텍트 **24.10.1~25.06**

Trident Protect 24.10.1부터 Trident Protect는 데이터 보호 작업 중에 KubeVirt 파일 시스템을 자동으로 동결 및 동결 해제합니다. 선택적으로 다음 명령을 사용하여 이 자동 동작을 비활성화할 수 있습니다.

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

▼ **Trident** 프로텍트 **24.10**

Trident Protect 24.10은 데이터 보호 작업 중에 KubeVirt VM 파일 시스템의 일관된 상태를 자동으로 보장하지 않습니다. Trident Protect 24.10을 사용하여 KubeVirt VM 데이터를 보호하려면 데이터 보호 작업을 시작하기 전에 파일 시스템의 동결/동결 해제 기능을 수동으로 활성화해야 합니다. 이렇게 하면 파일 시스템이 일관된 상태를 유지하게 됩니다.

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
```

Trident Protect를 사용하여 애플리케이션을 정의하고 VM을 복원하는 방법을 알아보려면 "KubeVirt 가상 머신을 Trident Protect로 보호합니다"을 참조하십시오.

SnapMirror 복제에 대한 요구 사항

NetApp SnapMirror 복제는 다음 ONTAP 솔루션에 대해 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 Select를 참조하십시오

- NetApp Cloud Volumes ONTAP를 참조하십시오
- NetApp ONTAP용 Amazon FSx

SnapMirror 복제를 위한 ONTAP 클러스터 요구 사항

SnapMirror 복제를 사용하려는 경우 ONTAP 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- * NetApp Trident*: NetApp Trident 백엔드로 ONTAP 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 존재해야 합니다. Trident Protect는 다음 드라이버로 지원되는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 복제를 지원합니다.
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san` : NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기 라이센스를 활성화해야 합니다. 자세한 내용은 을 "ONTAP의 SnapMirror 라이센스 개요" 참조하십시오.

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

참고	SnapMirror 비동기 보호만 지원됩니다.

SnapMirror 복제에 대한 피어링 고려 사항

스토리지 백엔드 피어링을 사용하려는 경우 환경이 다음 요구 사항을 충족하는지 확인하십시오.

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

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

- **NetApp Trident** 및 **SVM**: 피어링된 원격 SVM은 대상 클러스터의 NetApp Trident에서 사용할 수 있어야 합니다.
- 관리형 백엔드: 복제 관계를 생성하려면 Trident Protect에서 ONTAP 스토리지 백엔드를 추가하고 관리해야 합니다.

SnapMirror 복제를 위한 Trident/ONTAP 구성

Trident Protect를 사용하려면 소스 및 대상 클러스터 모두에 대한 복제를 지원하는 하나 이상의 스토리지 백엔드를 구성해야 합니다. 소스 및 대상 클러스터가 동일한 경우, 최상의 복원력을 위해 대상 애플리케이션은 소스 애플리케이션과 다른 스토리지 백엔드를 사용해야 합니다.

SnapMirror 복제를 위한 Kubernetes 클러스터 요구 사항

Kubernetes 클러스터가 다음 요구 사항을 충족하는지 확인하세요.

- **AppVault** 접근성: 소스 클러스터와 대상 클러스터 모두 애플리케이션 개체 복제를 위해 AppVault에서 읽고 쓸 수 있는 네트워크 액세스 권한이 있어야 합니다.

- 네트워크 연결: WAN을 통해 클러스터와 AppVault 간의 통신을 활성화하기 위해 방화벽 규칙, 버킷 권한 및 IP 허용 목록을 구성합니다.

참고	많은 기업 환경에서는 WAN 연결 전반에 걸쳐 엄격한 방화벽 정책을 구현합니다. 복제를 구성하기 전에 인프라 팀과 함께 이러한 네트워크 요구 사항을 확인하세요.

Trident Protect 설치 및 구성

사용자 환경이 Trident Protect에 대한 요구 사항을 충족하는 경우 다음 단계에 따라 클러스터에 Trident Protect를 설치할 수 있습니다. NetApp 에서 Trident Protect를 구입하거나 개인 레지스트리에서 설치할 수 있습니다. 클러스터가 인터넷에 액세스할 수 없는 경우 개인 레지스트리에서 설치하는 것이 도움이 됩니다.

Trident Protect 설치

NetApp 에서 Trident Protect 설치

단계

1. Trident Helm 저장소 추가:

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

2. Helm을 사용하여 Trident Protect를 설치하세요. 바꾸다 `<name-of-cluster>` 클러스터에 할당되고 클러스터의 백업과 스냅샷을 식별하는 데 사용되는 클러스터 이름이 포함됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect
```

3. 선택적으로, 디버그 로깅을 활성화하려면(문제 해결에 권장됨) 다음을 사용합니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect
```

   디버그 로깅은 NetApp 이 로그 수준을 변경하거나 문제를 재현하지 않고도 문제를 해결하는 데 도움이 됩니다.

개인 레지스트리에서 Trident Protect 설치

Kubernetes 클러스터가 인터넷에 액세스할 수 없는 경우 개인 이미지 레지스트리에서 Trident Protect를 설치할 수 있습니다. 다음 예에서 괄호 안의 값을 사용자 환경의 정보로 바꾸세요.

단계

1. 다음 이미지를 로컬 컴퓨터로 가져와서 태그를 업데이트한 다음 개인 레지스트리에 푸시합니다.

```
docker.io/netapp/controller:26.10.1
docker.io/netapp/trident-protect-utils:v6.0.0
docker.io/netapp/trident-protect-guestfs:v2.0.0
docker.io/netapp/kopiablockrestore:26.10.1
docker.io/netapp/trident-autosupport:26.10.1
docker.io/netapp/exechook:26.10.1
docker.io/netapp/resourcebackup:26.10.1
docker.io/netapp/resourcerestore:26.10.1
docker.io/netapp/resourcedelete:26.10.1
docker.io/netapp/filebrowse-transfer:26.10.1
```

   예를 들면 다음과 같습니다.

```
docker pull docker.io/netapp/controller:26.10.1
```

```
docker tag docker.io/netapp/controller:26.10.1 <private-registry-url>/controller:26.10.1
```

```
docker push <private-registry-url>/controller:26.10.1
```

   참고 Helm 차트를 얻으려면 먼저 인터넷에 연결된 컴퓨터에서 `helm pull trident-protect --version 100.2610.0 --repo https://netapp.github.io/trident-protect-helm-chart`를 사용하여 Helm 차트를 다운로드한 다음, 생성된 `trident-protect-100.2610.0.tgz` 파일을 오프라인 환경으로 복사하고 마지막 단계에서 저장소 참조 대신 `helm install trident-protect ./trident-protect-100.2610.0.tgz`를 사용하여 설치하십시오.

2. Trident Protect 시스템 네임스페이스를 만듭니다.

```
kubectl create ns trident-protect
```

3. 레지스트리에 로그인합니다.

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 개인 레지스트리 인증에 사용할 풀 암호를 만듭니다.

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Trident Helm 저장소 추가:

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

6. 라는 이름의 파일을 만듭니다. `protectValues.yaml` . 다음 Trident Protect 설정이 포함되어 있는지 확인하세요.

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

참고 그만큼 `imageRegistry` 그리고 `imagePullSecrets` 값은 다음을 포함한 모든 구성 요소 이미지에 적용됩니다. `resourcebackup` 그리고 `resourcerestore` . 레지스트리 내의 특정 저장소 경로에 이미지를 푸시하는 경우(예: `example.com:443/my-repo`), 레지스트리 필드에 전체 경로를 포함합니다. 이렇게 하면 모든 이미지가 다음에서 가져오게 됩니다. `<private-registry-url>/<image-name>:<tag>`.

7. Helm을 사용하여 Trident Protect를 설치하세요. 바꾸다 `<name_of_cluster>` 클러스터에 할당되고 클러스터의 백업과 스냅샷을 식별하는 데 사용되는 클러스터 이름이 포함됩니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. 선택적으로, 디버그 로깅을 활성화하려면(문제 해결에 권장됨) 다음을 사용합니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

디버그 로깅은 NetApp 이 로그 수준을 변경하거나 문제를 재현하지 않고도 문제를 해결하는 데 도움이 됩니다.

참고 AutoSupport 설정 및 네임스페이스 필터링을 포함한 추가 Helm 차트 구성 옵션은 다음을 참조하세요.

"Trident Protect 설치 사용자 정의".

Trident Protect CLI 플러그인을 설치하세요

Trident Protect 명령줄 플러그인을 사용할 수 있습니다. 이 플러그인은 Trident 의 확장 기능입니다. `tridentctl` Trident Protect 사용자 정의 리소스(CR)를 생성하고 상호 작용할 수 있는 유틸리티입니다.

Trident Protect CLI 플러그인을 설치하세요

명령줄 유틸리티를 사용하기 전에 클러스터에 액세스하는 데 사용하는 시스템에 설치해야 합니다. 컴퓨터에서 x64 또는 ARM CPU를 사용하는지 여부에 따라 다음 단계를 수행합니다.

Linux AMD64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-linux-amd64
```

Linux ARM64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-linux-arm64
```

Mac AMD64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-amd64
```

Mac ARM64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인을 다운로드하세요:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-arm64
```

1. 플러그인 바이너리에 대한 실행 권한 활성화:

```
chmod +x tridentctl-protect
```

2. PATH 변수에 정의된 위치에 플러그인 바이너리를 복사합니다. 예를 `/usr/bin` 들어, 또는 `/usr/local/bin` (Elevated Privileges가 필요할 수 있음):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 선택적으로 플러그인 바이너리를 홈 디렉토리의 위치로 복사할 수 있습니다. 이 경우 위치가 PATH 변수의 일부인지 확인하는 것이 좋습니다.

```
cp ./tridentctl-protect ~/bin/
```

참고 PATH 변수의 위치에 플러그인을 복사하면 또는 `tridentctl protect` 임의의 위치에서 플러그인을 사용할 수 `tridentctl-protect` 있습니다.

Trident CLI 플러그인 도움말을 봅니다

내장 플러그인 도움말 기능을 사용하여 플러그인 기능에 대한 자세한 도움말을 얻을 수 있습니다.

단계

1. 도움말 기능을 사용하여 사용 지침을 봅니다.

```
tridentctl-protect help
```

명령 자동 완성 활성화

Trident Protect CLI 플러그인을 설치한 후 특정 명령에 대한 자동 완성을 활성화할 수 있습니다.

Bash 셸에 대해 자동 완성 기능을 활성화합니다

단계

1. 완료 스크립트를 만듭니다.

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. 스크립트를 포함할 홈 디렉토리에 새 디렉토리를 만듭니다.

```
mkdir -p ~/.bash/completions
```

3. 다운로드한 스크립트를 디렉터리로 이동합니다 `~/.bash/completions`.

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 홈 디렉토리의 파일에 다음 줄을 추가합니다 `~/.bashrc`.

```
source ~/.bash/completions/tridentctl-completion.bash
```

Z 쉘에 대한 자동 완성 기능을 활성화합니다

단계

1. 완료 스크립트를 만듭니다.

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. 스크립트를 포함할 홈 디렉토리에 새 디렉토리를 만듭니다.

```
mkdir -p ~/.zsh/completions
```

3. 다운로드한 스크립트를 디렉터리로 이동합니다 `~/.zsh/completions`.

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 홈 디렉토리의 파일에 다음 줄을 추가합니다 `~/.zprofile`.

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

결과

다음 쉘 로그인 시 tridentctl-protect 플러그인으로 명령 자동 완성 기능을 사용할 수 있습니다.

Trident Protect 설치 사용자 정의

Trident Protect의 기본 구성을 사용자 환경의 특정 요구 사항에 맞게 사용자 정의할 수 있습니다.

Trident Protect 컨테이너 리소스 제한 지정

Trident Protect를 설치한 후에는 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 리소스 제한을 지정할 수 있습니다. 리소스 제한을 설정하면 Trident Protect 작업에서 클러스터 리소스가 얼마나 소모되는지 제어할 수 있습니다.

단계

1. 라는 이름의 파일을 `resourceLimits.yaml` 만듭니다.
2. 사용자 환경의 요구 사항에 따라 Trident Protect 컨테이너에 대한 리소스 제한 옵션으로 파일을 채웁니다.

   다음 예제 구성 파일은 사용 가능한 설정을 보여 주며 각 리소스 제한에 대한 기본값을 포함합니다.

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
```

```
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. 파일의 값을 적용합니다 `resourceLimits.yaml`.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

보안 컨텍스트 제약 조건을 사용자 지정합니다

Trident Protect를 설치한 후에는 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 OpenShift 보안 컨텍스트 제약 조건(SCC)을 수정할 수 있습니다. 이러한 제약 조건은 Red Hat OpenShift 클러스터의 포드에 대한 보안 제한 사항을 정의합니다.

단계

1. 라는 이름의 파일을 `sccconfig.yaml` 만듭니다.
2. 파일에 SCC 옵션을 추가하고 사용자 환경의 필요에 따라 매개 변수를 수정합니다.

   다음 예는 SCC 옵션에 대한 매개 변수의 기본값을 보여줍니다.

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

   다음 표에서는 SCC 옵션에 대한 매개 변수를 설명합니다.

매개 변수	설명	기본값
생성	SCC 자원을 생성할 수 있는지 여부를 결정한다. SCC 리소스는 가로 설정되어 `true` 있고 Helm 설치 프로세스에서 OpenShift 환경을 식별하는 경우에만 `scc.create` 생성됩니다. OpenShift에서 작동하지 않거나 이 로 설정된 `false` 경우 `scc.create` SCC 리소스가 생성되지 않습니다.	참
이름	SCC의 이름을 지정합니다.	Trident-protect-job을 선택합니다

매개 변수	설명	기본값
우선 순위	SCC의 우선 순위를 정의합니다. 우선 순위가 높은 SCC는 값이 낮은 SCC보다 먼저 평가됩니다.	1

3. 파일의 값을 적용합니다 `sccconfig.yaml`.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

그러면 기본값이 파일에 지정된 값으로 `sccconfig.yaml` 바뀝니다.

추가 **Trident Protect** 헬름 차트 설정 구성

AutoSupport 설정, 네임스페이스 필터링 및 OpenShift 웹훅 인증서 관리를 특정 요구 사항에 맞게 사용자 지정할 수 있습니다. 다음 표는 사용 가능한 구성 매개변수를 설명합니다.

매개 변수	유형	설명
autoSupport.proxy	문자열	NetApp AutoSupport 연결을 위한 프록시 URL을 구성합니다. 이를 사용하면 프록시 서버를 통해 지원 번들 업로드를 라우팅할 수 있습니다. 예: `http://my.proxy.url` .
autoSupport.안전하지 않음	부울	AutoSupport 프록시 연결에 대한 TLS 검증을 건너뜁니다. `true` . 안전하지 않은 프록시 연결에만 사용하세요. (기본: `false`)
autoSupport.활성화됨	부울	일일 Trident Protect AutoSupport 번들 업로드를 활성화하거나 비활성화합니다. 설정 시 `false` , 예약된 일일 업로드는 비활성화되지만, 여전히 수동으로 지원 번들을 생성할 수 있습니다. (기본: `true`)
restoreSkipNamespaceAnnotations	문자열	백업 및 복원 작업에서 제외할 네임스페이스 주석의 쉼표로 구분된 목록입니다. 주석을 기준으로 네임스페이스를 필터링할 수 있습니다.
restoreSkipNamespaceLabels	문자열	백업 및 복원 작업에서 제외할 네임스페이스 레이블의 쉼표로 구분된 목록입니다. 라벨을 기준으로 네임스페이스를 필터링할 수 있습니다.

매개 변수	유형	설명
useOpenshiftServiceCA	부울	OpenShift의 기본 서비스 CA Operator와의 통합을 활성화하여 Trident Protect 승인 웹훅에 대한 TLS 인증서를 자동으로 생성하고 주입합니다. 이 옵션을 `true`로 설정하면 서비스 CA Operator가 웹훅 TLS 인증서를 자동으로 관리합니다. OpenShift 클러스터에서만 사용하십시오. (기본값: `false)

스냅샷 및 백업 메타데이터를 암호화하려면 "Trident Protect 스냅샷 및 백업용 AppVault 메타데이터 암호화"를 참조하십시오.

YAML 구성 파일이나 명령줄 플래그를 사용하여 이러한 옵션을 구성할 수 있습니다.

YAML 파일 사용

단계

1. 구성 파일을 만들고 이름을 지정하세요. `values.yaml` .
2. 생성한 파일에 사용자 정의하려는 구성 옵션을 추가합니다.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
useOpenshiftServiceCA: true
```

3. 다음을 채운 후 `values.yaml` 올바른 값을 가진 파일을 만들려면 구성 파일을 적용하세요:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

CLI 플래그 사용

단계

1. 다음 명령을 사용하십시오. `--set` 개별 매개변수를 지정하는 플래그:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
```

```
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --set useOpenshiftServiceCA=true \
  --reuse-values
```

Trident Protect 포드를 특정 노드로 제한

Kubernetes nodeSelector 노드 선택 제약 조건을 사용하면 노드 레이블을 기준으로 Trident Protect 포드를 실행할 수 있는 노드를 제어할 수 있습니다. 기본적으로 Trident Protect는 Linux를 실행하는 노드로 제한됩니다. 사용자의 요구 사항에 따라 이러한 제약 조건을 추가로 사용자 정의할 수 있습니다.

단계

1. 라는 이름의 파일을 `nodeSelectorConfig.yaml` 만듭니다.
2. 파일에 nodeSelector 옵션을 추가하고 해당 파일을 수정하여 노드 레이블을 추가하거나 변경하여 환경 요구 사항에 따라 제한합니다. 예를 들어, 다음 파일에는 기본 OS 제한이 포함되어 있지만 특정 지역 및 앱 이름도 대상으로 합니다.

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. 파일의 값을 적용합니다 `nodeSelectorConfig.yaml`.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

이렇게 하면 기본 제한 사항이 파일에 지정한 제한 사항으로 `nodeSelectorConfig.yaml` 바뀝니다.

Trident Protect 관리

Trident Protect 권한 및 액세스 제어 관리

Trident Protect는 역할 기반 액세스 제어(RBAC)의 Kubernetes 모델을 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스와 연관된 기본 서비스 계정을 제공합니다. 사용자 수가 많거나 특정 보안 요구 사항이 있는 조직인 경우 Trident Protect의 RBAC 기능을 사용하여 리소스와 네임스페이스에 대한 액세스를 보다 세부적으로 제어할 수 있습니다.

클러스터 관리자는 항상 기본 네임스페이스의 리소스에 액세스할 수 `trident-protect` 있으며 다른 모든 네임스페이스의 리소스에 액세스할 수도 있습니다. 리소스 및 응용 프로그램에 대한 액세스를 제어하려면 추가 네임스페이스를 만들고 해당 네임스페이스에 리소스 및 응용 프로그램을 추가해야 합니다.

사용자는 기본 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만들 수 `trident-protect` 없습니다. 응용 프로그램 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만들어야 합니다(모범 사례로서 연결된 응용 프로그램과 동일한 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만듭니다).

참고

다음을 포함하는 권한이 있는 Trident Protect 사용자 정의 리소스 개체에 대한 액세스 권한은 관리자에게만 부여됩니다.

- * AppVault *: 버킷 자격 증명 데이터가 필요합니다
- **AutoSupportBundle**: 메트릭, 로그 및 기타 중요한 Trident Protect 데이터를 수집합니다.
- * AutoSupport기술에 따라 로그 수집 일정을 관리할 수 있습니다

가장 좋은 방법은 RBAC를 사용하여 권한이 있는 객체에 대한 액세스를 관리자에게 제한하는 것입니다.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 를 "Kubernetes RBAC 설명서"참조하십시오.

서비스 계정에 대한 자세한 내용은 을 "Kubernetes 서비스 계정 설명서"참조하십시오.

예**:** 두 사용자 그룹에 대한 액세스를 관리합니다

예를 들어, 조직에는 클러스터 관리자, 엔지니어링 사용자 그룹 및 마케팅 사용자 그룹이 있습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각각 해당 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 만들기 위해 다음 작업을 완료합니다.

1단계**:** 각 그룹의 리소스를 포함할 네임스페이스를 만듭니다

네임스페이스를 만들면 리소스를 논리적으로 분리하고 해당 리소스에 액세스할 수 있는 사용자를 보다 효율적으로 제어할 수 있습니다.

단계

1. 엔지니어링 그룹의 네임스페이스를 만듭니다.

```
kubectl create ns engineering-ns
```

2. 마케팅 그룹의 네임스페이스를 만듭니다.

```
kubectl create ns marketing-ns
```

2단계**:** 각 네임스페이스의 리소스와 상호 작용할 새 서비스 계정을 만듭니다

새로 만드는 각 네임스페이스에는 기본 서비스 계정이 함께 제공되지만, 나중에 필요한 경우 그룹 간에 Privileges를 추가로 나눌 수 있도록 각 사용자 그룹에 대한 서비스 계정을 만들어야 합니다.

단계

1. 엔지니어링 그룹에 대한 서비스 계정을 생성합니다.

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 마케팅 그룹의 서비스 계정 만들기:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

3단계**:** 각 새 서비스 계정에 대한 암호를 만듭니다

서비스 계정 암호는 서비스 계정을 인증하는 데 사용되며, 손상된 경우 쉽게 삭제하고 다시 만들 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 대한 암호를 생성합니다.

```yaml
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 암호 만들기:

```yaml
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 개체를 만들어 ClusterRole 개체를 각 새 서비스 계정에 바인딩합니다

Trident Protect를 설치하면 기본 ClusterRole 개체가 생성됩니다. RoleBinding 객체를 생성하고 적용하여 이 ClusterRole을 서비스 계정에 바인딩할 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 연결:

```
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: 권한을 테스트합니다

권한이 올바른지 테스트합니다.

단계

1. 엔지니어링 사용자가 엔지니어링 리소스에 액세스할 수 있는지 확인합니다.

```
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 객체에 대한 액세스 권한 부여

백업 및 스냅샷과 같은 데이터 관리 작업을 수행하려면 클러스터 관리자가 개별 사용자에게 AppVault 개체에 대한 액세스 권한을 부여해야 합니다.

단계

1. AppVault에 대한 사용자 액세스 권한을 부여하는 AppVault 및 암호 조합 YAML 파일을 만들고 적용합니다. 예를 들어, 다음 CR은 사용자에게 AppVault에 대한 액세스 권한을 `eng-user` 부여합니다.

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
```

```
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. 클러스터 관리자가 네임스페이스의 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 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 만들고 적용하여 권한을 사용자 eng-user에 바인딩합니다. 예를 들면 다음과 같습니다.

```
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 개체 정보를 검색하려고 합니다.

   ```
   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 얻을 수 있는지 테스트해 봅니다.

   ```
   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 부여한 사용자는 응용 프로그램 데이터 관리 작업에 승인된 AppVault 개체를 사용할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 생성할 수 없습니다.

Trident Protect 리소스 모니터링

kube-state-metrics, Prometheus, Alertmanager 오픈 소스 도구를 사용하여 Trident Protect로 보호되는 리소스의 상태를 모니터링할 수 있습니다.

kube-state-metrics 서비스는 Kubernetes API 통신에서 메트릭을 생성합니다. Trident Protect와 함께 사용하면 사용자 환경의 리소스 상태에 대한 유용한 정보가 제공됩니다.

Prometheus는 kube-state-metrics에서 생성된 데이터를 수집하고 이러한 객체에 대한 쉽게 읽을 수 있는 정보로 제공할 수 있는 툴킷입니다. kube-state-metrics와 Prometheus를 함께 사용하면 Trident Protect로 관리하는 리소스의 상태와 상태를 모니터링할 수 있는 방법을 제공합니다.

Alertmanager는 Prometheus와 같은 도구에서 보낸 경고를 수집하여 구성한 대상으로 라우팅하는 서비스입니다.

참고

이 단계에 포함된 구성 및 지침은 예일 뿐입니다. 환경에 맞게 사용자 지정해야 합니다. 구체적인 지침 및 지원은 다음 공식 문서를 참조하십시오.

- "Kudbe-state-metrics 문서"
- "Prometheus 설명서"
- "Alertmanager 설명서"

1단계**:** 모니터링 도구를 설치합니다

Trident Protect에서 리소스 모니터링을 활성화하려면 kube-state-metrics, Promethus, Alertmanager를 설치하고 구성해야 합니다.

kube-state-metrics를 설치합니다

Helm을 사용하여 kube-state-metrics를 설치할 수 있습니다.

단계

1. kube-state-metrics Helm 차트를 추가합니다. 예를 들면 다음과 같습니다.

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. 클러스터에 Prometheus ServiceMonitor CRD를 적용합니다.

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Helm 차트에 대한 구성 파일을 만듭니다(예: `metrics-config.yaml`). 다음 예제 구성을 사용자 환경에 맞게 사용자 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 설치합니다. 예를 들면 다음과 같습니다.

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Trident Protect에서 사용하는 사용자 정의 리소스에 대한 메트릭을 생성하도록 kube-state-metrics를 구성하려면 다음 지침을 따르세요. "kube-state-metrics 맞춤형 리소스 문서화" .

Prometheus를 설치합니다

의 지침에 따라 Prometheus를 설치할 수 "Prometheus 설명서" 있습니다.

Alertmanager를 설치합니다

의 지침에 따라 Alertmanager를 설치할 수 "Alertmanager 설명서" 있습니다.

2단계**:** 함께 작동하도록 모니터링 도구를 구성합니다

모니터링 도구를 설치한 후에는 함께 작동하도록 구성해야 합니다.

단계

1. kube-state-metrics를 Prometheus와 통합합니다. Prometheus 구성 파일을 (`prometheus.yaml`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. 알림을 Alertmanager로 라우팅하도록 Prometheus를 구성합니다. Prometheus 구성 파일을 (`prometheus.yaml`편집하고 다음 섹션을 추가합니다.

   prometheus.yaml: Alertmanager에 알림 보내기

   ```
   alerting:
     alertmanagers:
       - static_configs:
           - targets:
               - alertmanager.trident-protect.svc:9093
   ```

결과

Prometheus는 이제 kube 상태 메트릭에서 메트릭을 수집하고 Alertmanager에 경고를 전송할 수 있습니다. 이제 알림을 트리거하는 조건과 알림을 보낼 위치를 구성할 준비가 되었습니다.

3단계**:** 경고 및 경고 대상을 구성합니다

함께 작동하도록 도구를 구성한 후에는 알림을 트리거할 정보 유형과 알림을 보낼 위치를 구성해야 합니다.

경고 예**:** 백업 실패

다음 예에서는 백업 사용자 지정 리소스의 상태가 5초 이상 으로 설정된 경우 트리거되는 중요 알림을 `Error` 정의합니다. 이 예제를 사용자 환경에 맞게 사용자 지정하고 이 YAML 스니펫을 구성 파일에 포함할 수

`prometheus.yaml` 있습니다.

rules.yaml: 실패한 백업에 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."
```

알림을 다른 채널로 보내도록 **Alertmanager**를 구성합니다

파일에 각 구성을 지정하여 전자 메일, PagerDuty, Microsoft Teams 또는 기타 알림 서비스와 같은 다른 채널에 알림을 보내도록 Alertmanager를 구성할 수 `alertmanager.yaml` 있습니다.

다음 예에서는 알림 메시지를 Slack 채널에 보내도록 Alertmanager를 구성합니다. 이 예제를 환경에 맞게 사용자 지정하려면 키 값을 사용자 환경에서 사용되는 Slack Webhook URL로 바꿉니다 `api_url`.

alertmanager.yaml: Slack 채널에 알림 보내기

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Trident Protect 지원 번들 생성

Trident Protect를 사용하면 관리자는 관리 중인 클러스터와 앱에 대한 로그, 메트릭, 토폴로지 정보 등 NetApp 지원에 유용한 정보가 포함된 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 사용자 정의 리소스(CR) 파일을 사용하여 NetApp 지원 사이트(NSS)에 지원 번들을 업로드할 수 있습니다.

CR을 사용하여 지원 번들을 만듭니다

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml`).
2. 다음 특성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.triggerType *:(*required*) 지원 번들이 즉시 생성되는지 또는 예약되는지 여부를 결정합니다. 예약된 번들 생성은 UTC 오전 12시에 수행됩니다. 가능한 값:
     - 예약됨
     - 수동
   - **spec.uploadEnabled**: (*Optional*) 지원 번들이 생성된 후 NetApp 지원 사이트에 업로드되어야 하는지 여부를 제어합니다. 지정하지 않으면 가 기본값으로 `false` 설정됩니다. 가능한 값:
     - 참
     - FALSE(기본값)
   - **spec.dataWindowStart**: (*Optional*) 지원 번들에 포함된 데이터의 창이 시작되는 날짜와 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않을 경우 기본값은 24시간 이전입니다. 지정할 수 있는 가장 빠른 기간 날짜는 7일 이전입니다.

     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. 다음을 채운 후 `trident-protect-support-bundle.yaml` 올바른 값으로 파일을 만들고 CR을 적용하세요.

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

CLI를 사용하여 지원 번들을 생성합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 지원 번들을 만듭니다. 은 `trigger-type` 번들이 즉시

생성되는지 아니면 생성 시간이 스케줄에 따라 결정되는지, 또는 `Scheduled` 이 될 수 있는지 결정합니다 `Manual`. 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 만든 후에는 생성 진행 상황을 모니터링하고 로컬 시스템으로 검색할 수 있습니다.

단계

1. 기다리다 `status.generationState` 도달하다 `Completed` 상태. 다음 명령을 사용하여 생성 진행 상황을 모니터링할 수 있습니다.

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-
protect
```

2. 지원 번들을 로컬 시스템으로 가져옵니다. 완성된 AutoSupport 번들에서 복사 명령을 가져옵니다.

```
kubectl describe autosupportbundle trident-protect-support-bundle -n
trident-protect
```

찾아라 `kubectl cp` 출력에서 명령을 선택하여 실행하고, 대상 인수를 원하는 로컬 디렉토리로 바꿉니다.

Trident 프로텍트 업그레이드

Trident Protect를 최신 버전으로 업그레이드하면 새로운 기능이나 버그 수정을 활용할 수 있습니다.

참고

- 24.10 버전에서 업그레이드하는 경우, 업그레이드 중 실행 중인 스냅샷이 실패할 수 있습니다. 이 실패는 향후 수동 또는 예약된 스냅샷 생성을 방해하지 않습니다. 업그레이드 중 스냅샷이 실패할 경우, 수동으로 새 스냅샷을 생성하여 애플리케이션을 보호할 수 있습니다.

  잠재적인 오류를 방지하려면 업그레이드 전에 모든 스냅샷 일정을 비활성화했다가 나중에 다시 활성화할 수 있습니다. 하지만 이렇게 하면 업그레이드 기간 동안 예약된 스냅샷이 모두 손실될 수 있습니다.

- 개인 레지스트리 설치의 경우 대상 버전에 필요한 Helm 차트와 이미지가 개인 레지스트리에서 사용 가능한지 확인하고, 사용자 지정 Helm 값이 새 차트 버전과 호환되는지 확인하세요. 자세한 내용은 다음을 참조하세요."개인 레지스트리에서 Trident Protect 설치" .

1단계: 버전을 선택합니다

Trident Protect 버전은 날짜 기반 `YY.MM` 명명 규칙을 따르며, 여기서 "YY"는 연도의 마지막 두 자리 숫자이고 "MM"은 월을 나타냅니다. 도트 릴리스는 `YY.MM.X` 규칙을 따르며, 여기서 "X"는 패치 레벨을 나타냅니다. 업그레이드할 버전은 업그레이드하려는 현재 버전을 기준으로 선택합니다.

- 설치된 버전에서 4개 릴리스 이내의 대상 릴리스로 직접 업그레이드할 수 있습니다. 예를 들어 24.10(또는 24.10의 모든 마이너 버전)에서 25.10으로 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, 24.10 버전을 실행 중이고 26.10 버전으로 업그레이드하려는 경우:
  a. 먼저 24.10에서 25.06으로 업그레이드하십시오.
  b. 그다음 25.06에서 26.10으로 업그레이드하세요.

2단계: **Trident Protect** 업그레이드

Trident Protect를 업그레이드하려면 다음 단계를 수행하세요.

단계

1. Trident Helm 리포지토리를 업데이트합니다.

```
helm repo update
```

2. Trident Protect CRD 업그레이드:

참고	25.06 이전 버전에서 업그레이드하는 경우 CRD가 이제 Trident Protect Helm 차트에 포함되었으므로 이 단계가 필요합니다.

   a. CRD 관리를 전환하려면 이 명령을 실행하세요. `trident-protect-crds` 에게 `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

   b. Helm 비밀을 삭제하려면 이 명령을 실행하세요. `trident-protect-crds` 차트:

참고	제거하지 마십시오 `trident-protect-crds` Helm을 사용하여 차트를 만들면 CRD와 관련 데이터가 제거될 수 있습니다.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident 프로텍트 업그레이드:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2610.0 --namespace trident-protect
```

참고 | 업그레이드 중에 로깅 수준을 구성하려면 다음을 추가하세요. `--set logLevel=debug` 업그레이드 명령으로. 기본 로깅 수준은 다음과 같습니다. `warn` . 디버그 로깅은 NetApp 지원팀이 로그 수준을 변경하거나 문제를 재현하지 않고도 문제를 진단하는 데 도움이 되므로 문제 해결에 권장됩니다.

애플리케이션을 관리하고 보호합니다

Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다.

Trident Protect의 버킷 사용자 정의 리소스(CR)는 AppVault로 알려져 있습니다. AppVault 객체는 스토리지 버킷의 선언적 Kubernetes 워크플로 표현입니다. AppVault CR에는 백업, 스냅샷, 복원 작업, SnapMirror 복제와 같은 보호 작업에 버킷을 사용하는 데 필요한 구성이 포함되어 있습니다. 관리자만 AppVault를 만들 수 있습니다.

애플리케이션에서 데이터 보호 작업을 수행할 때는 AppVault CR을 수동으로 또는 명령줄에서 생성해야 합니다. AppVault CR은 사용자 환경에 따라 다르므로, 이 페이지의 예시를 참고하여 AppVault CR을 생성할 수 있습니다.

참고 | Trident Protect가 설치된 클러스터에 AppVault CR이 있는지 확인하세요. AppVault CR이 없거나 액세스할 수 없는 경우 명령줄에 오류가 표시됩니다.

AppVault 인증 및 암호를 구성합니다

AppVault CR을 생성하기 전에 AppVault와 선택한 데이터 이동자가 공급자 및 관련 리소스에 대해 인증할 수 있는지 확인하세요.

Data Mover 저장소 암호입니다

CR이나 Trident Protect CLI 플러그인을 사용하여 AppVault 객체를 생성하는 경우 Restic 및 Kopia 암호화를 위한 사용자 정의 비밀번호가 포함된 Kubernetes 비밀번호를 지정할 수 있습니다. 비밀번호를 지정하지 않으면 Trident Protect는 기본 비밀번호를 사용합니다.

- AppVault CR을 수동으로 생성할 때 **spec.dataMoverPasswordSecretRef** 필드를 사용하여 비밀번호를 지정합니다.
- Trident Protect CLI를 사용하여 AppVault 객체를 생성할 때 다음을 사용하십시오. `--data-mover-password-secret-ref` 비밀을 지정하는 인수입니다.

참고 | 스냅샷 및 백업 메타데이터를 암호화하거나 볼트 암호를 변경하려면 `spec.vaultPasswordSecretRef` 필드를 사용하십시오. "Trident Protect 스냅샷 및 백업용 AppVault 메타데이터 암호화"를 참조하십시오.

Data Mover 리포지토리 암호 암호를 생성합니다

다음 예를 사용하여 비밀번호를 생성하세요. AppVault 객체를 생성할 때 Trident Protect가 이 비밀번호를 사용하여

데이터 이동 저장소를 인증하도록 지시할 수 있습니다.

참고

- 사용 중인 Data Mover에 따라 해당 Data Mover에 대한 암호만 포함하면 됩니다. 예를 들어, Restic을 사용하고 있고 나중에 Kopia를 사용할 계획이 없는 경우 암호를 만들 때 Restic 암호만 포함할 수 있습니다.
- 비밀번호를 안전한 곳에 보관하세요. 같은 클러스터나 다른 클러스터에서 데이터를 복원할 때 필요합니다. 클러스터 또는 `trident-protect` 네임스페이스가 삭제되면 비밀번호 없이는 백업이나 스냅샷을 복원할 수 없습니다.

CR을 사용합니다

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

CLI를 사용합니다

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

S3 호환 스토리지 IAM 권한

Amazon S3, Generic S3 등 S3 호환 스토리지에 접속할 경우 "StorageGRID S3" , 또는 "ONTAP S3 를 참조하십시오" Trident Protect를 사용하는 경우, 제공하는 사용자 자격 증명에 버킷에 액세스하는 데 필요한 권한이 있는지 확인해야 합니다. 다음은 Trident Protect에 대한 액세스에 필요한 최소한의 권한을 부여하는 정책의 예입니다. 이 정책은 S3 호환 버킷 정책을 관리하는 사용자에게 적용할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
```

```
            "s3:ListBucket",
            "s3:DeleteObject"
        ],
        "Resource": "*"
    }
  ]
}
```

Amazon S3 정책에 대한 자세한 내용은 다음 예제를 참조하세요. "Amazon S3 문서" .

Amazon S3(AWS) 인증을 위한 EKS Pod ID

Trident Protect는 Kopia 데이터 이동 작업을 위해 EKS Pod Identity를 지원합니다. 이 기능을 사용하면 Kubernetes 비밀에 AWS 자격 증명을 저장하지 않고도 S3 버킷에 안전하게 액세스할 수 있습니다.

- Trident Protect를 사용한 EKS Pod Identity 요구 사항*

Trident Protect와 함께 EKS Pod Identity를 사용하기 전에 다음 사항을 확인하세요.

- EKS 클러스터에 Pod Identity가 활성화되어 있습니다.
- 필요한 S3 버킷 권한이 있는 IAM 역할을 생성했습니다. 자세한 내용은 다음을 참조하세요."S3 호환 스토리지 IAM 권한" .
- IAM 역할은 다음 Trident Protect 서비스 계정과 연결됩니다.
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Pod Identity를 활성화하고 IAM 역할을 서비스 계정과 연결하는 방법에 대한 자세한 지침은 다음을 참조하세요. "AWS EKS Pod Identity 문서" .

AppVault 구성 EKS Pod Identity를 사용하는 경우 AppVault CR을 다음과 같이 구성합니다. `useIAM: true` 명시적 자격 증명 대신 플래그:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
```

```
      useIAM: true
```

클라우드 공급자를 위한 **AppVault** 키 생성 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 예입니다. 다음 예를 사용하여 각 클라우드 공급자의 자격 증명에 대한 키를 생성할 수 있습니다.

Google 클라우드

```
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 Azure를 참조하십시오

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

일반 **S3**

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3 를 참조하십시오

```
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 예

다음은 각 공급자에 대한 AppVault 정의의 예입니다.

AppVault CR 예

다음 CR 예제를 사용하여 각 클라우드 공급자에 대한 AppVault 개체를 만들 수 있습니다.

참고

- 선택적으로 Restic 및 Kopia 저장소 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 암호를 지정할 수 있습니다. 자세한 내용은 을 Data Mover 저장소 암호입니다 참조하십시오.
- Amazon S3(AWS) AppVault 객체의 경우 선택적으로 sessionToken을 지정할 수 있습니다. 이는 인증에 SSO(Single Sign-On)를 사용하는 경우에 유용합니다. 이 토큰은 에서 공급자에 대한 키를 생성할 때 클라우드 공급자를 위한 AppVault 키 생성 예생성됩니다.
- S3 AppVault 객체의 경우 선택적으로 키를 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 `spec.providerConfig.S3.proxyURL` 있습니다.

Google 클라우드

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

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
```

참고

Kopia 데이터 무버와 함께 Pod Identity를 사용하는 EKS 환경의 경우 다음을 제거할 수 있습니다. `providerCredentials` 섹션을 추가하고 추가하세요 `useIAM: true` 아래에 `s3` 대신 구성을 선택하세요.

Microsoft Azure를 참조하십시오

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
```

```
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

일반 **S3**

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3 를 참조하십시오

```
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 예

다음 CLI 명령 예제를 사용하여 각 공급자에 대해 AppVault CRS를 만들 수 있습니다.

참고

- 선택적으로 Restic 및 Kopia 저장소 암호화에 대한 사용자 지정 암호가 포함된 Kubernetes 암호를 지정할 수 있습니다. 자세한 내용은 을 Data Mover 저장소 암호입니다 참조하십시오.

- S3 AppVault 개체의 경우 선택적으로 인수를 사용하여 아웃바운드 S3 트래픽에 대한 송신 프록시 URL을 지정할 수 `--proxy-url <ip_address:port>` 있습니다.

Google 클라우드

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

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 Azure를 참조하십시오

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

일반 **S3**

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3 를 참조하십시오

```
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
```

지원됨 **`providerConfig.s3`** 구성 옵션

S3 공급자 구성 옵션은 다음 표를 참조하십시오.

매개 변수	설명	기본값	예
`providerConfig.s3.skipCertValidation`	SSL/TLS 인증서 확인을 비활성화합니다.	거짓	"참", "거짓"
`providerConfig.s3.secure`	S3 엔드포인트와의 안전한 HTTPS 통신을 활성화합니다.	참	"참", "거짓"
`providerConfig.s3.proxyURL`	S3에 연결하는 데 사용되는 프록시 서버의 URL을 지정하십시오.	None	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	SSL/TLS 인증을 위해 사용자 지정 루트 CA 인증서를 제공하십시오.	None	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	S3 버킷에 접근하기 위해 IAM 인증을 활성화하세요. EKS Pod Identity에 적용됩니다.	거짓	참, 거짓

AppVault 정보를 봅니다

Trident Protect CLI 플러그인을 사용하면 클러스터에서 생성한 AppVault 개체에 대한 정보를 볼 수 있습니다.

단계

1. AppVault 개체의 내용을 봅니다.

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
```

```
-n trident-protect
```

◦ 출력 예 *:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   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 플래그를 `--show-paths`사용합니다.

   표의 첫 번째 열에 있는 클러스터 이름은 Trident Protect helm 설치 시 클러스터 이름이 지정된 경우에만 사용할 수 있습니다. 예를 들어: `--set clusterName=production1` .

AppVault를 제거합니다

언제든지 AppVault 개체를 제거할 수 있습니다.

참고	AppVault 개체를 삭제하기 전에 AppVault CR에서 키를 제거하지 `finalizers` 마십시오. 이렇게 하면 AppVault 버킷의 잔여 데이터와 클러스터의 분리된 리소스가 생성될 수 있습니다.

시작하기 전에

삭제하려는 AppVault에서 사용 중인 모든 스냅샷 및 백업 CRS를 삭제했는지 확인합니다.

Kubernetes CLI를 사용하여 **AppVault**를 제거합니다

1. AppVault 개체를 제거하고 `appvault-name` 제거할 AppVault 개체의 이름으로 바꿉니다.

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect CLI를 사용하여 **AppVault** 제거

1. AppVault 개체를 제거하고 `appvault-name` 제거할 AppVault 개체의 이름으로 바꿉니다.

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Trident Protect 스냅샷 및 백업용 AppVault 메타데이터 암호화

Kubernetes Secrets의 민감한 정보를 포함하여 Trident Protect가 AppVault에 저장하는 스냅샷 및 백업 메타데이터를 암호화할 수 있습니다. 암호화된 데이터를 보호하는 암호도 변경할 수 있습니다. 암호를 한 번만 변경하면 메타데이터 암호화 키가 재암호화되고 AppVault의 모든 Kopia 리포지토리와 모든 Restic 리포지토리가 새 암호로 이동됩니다.

암호화는 기본적으로 비활성화되어 있습니다. 암호화를 활성화하면 Trident Protect는 메타데이터를 AppVault에 저장하기 전에 새 스냅샷 및 백업 메타데이터를 암호화합니다. 이렇게 하면 애플리케이션 스냅샷 또는 백업에서 해당 리소스를 명시적으로 제외하지 않는 한, Secrets의 민감한 정보를 포함한 애플리케이션 리소스를 보호할 수 있습니다.

금고 암호는 데이터를 암호화하는 키를 보호합니다. 다음 동작에 유의하십시오.

- 금고 암호를 변경하면 Trident Protect는 새 암호로 동일한 키를 보호합니다. 이미 암호화된 데이터는 다시 기록하지 않습니다.
- 암호화를 활성화하기 전에 저장된 데이터는 계속 읽을 수 있습니다.
- 암호화된 데이터는 나중에 새 데이터에 대한 암호화를 해제하더라도 읽을 수 있는 상태로 유지됩니다.

이 작업에 대해

암호화를 활성화할 때, 사용자 지정 암호를 지정하지 않는 한 Trident Protect는 안전하지 않은 기본 암호를 사용합니다. 각 AppVault에 대해 사용자 지정 볼트 암호를 생성하고 `spec.vaultPasswordSecretRef`에서 이를 참조하십시오.

기본 암호로 암호화를 활성화한 후 나중에 사용자 지정 암호를 추가한 경우, 기존 데이터와 리포지토리가 사용자 지정 암호로 이동하도록 암호를 교체해야 합니다. 볼트 비밀번호를 변경하십시오을(를) 참조하십시오.

각 AppVault에 대해 고유한 사용자 지정 암호를 사용하십시오. 한 번에 하나의 AppVault에 대해서만 암호를 교체하고, 다른 암호 변경을 시작하기 전에 교체가 완료되었는지 확인하십시오.

시작하기 전에

다음 사항을 확인하십시오.

- `trident-protect` 네임스페이스에 대한 액세스.
- Kubernetes Secret을 생성하고 AppVault 사용자 정의 리소스(CR)에 패치를 적용할 수 있는 권한입니다.
- 각 AppVault에 대해 계획된 별도의 암호 Secret입니다.

메타데이터 암호화 활성화

Trident Protect에 대한 Helm 값을 설정하여 메타데이터 암호화를 활성화합니다.

단계

1. `resourceBackupEncryptionEnabled`를 `true`로 설정하려면 `values.yaml` 파일을 사용하십시오:
   a. 다음 내용을 `values.yaml` 파일에 추가하십시오:

   ```
   resourceBackupEncryptionEnabled: true
   ```

   b. 파일을 적용하십시오:

   ```
   helm upgrade trident-protect -n trident-protect netapp-trident-
   protect/trident-protect -f values.yaml --reuse-values
   ```

2. 또는 Helm `--set` 플래그를 사용하여 `resourceBackupEncryptionEnabled`을 `true`로 설정합니다.

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set resourceBackupEncryptionEnabled=true \
  --reuse-values
```

결과

Trident Protect는 새 스냅샷 및 백업 메타데이터를 암호화합니다.

다음 단계

각 AppVault에 대한 사용자 지정 볼트 암호를 생성하십시오. 생성하기 전까지 Trident Protect는 안전하지 않은 기본 암호를 사용합니다.

사용자 지정 볼트 암호 생성

사용자 지정 암호를 사용하려면 암호가 포함된 Kubernetes Secret을 생성한 다음 AppVault에서 Secret을 참조하십시오.

이 작업에 대해

AppVault를 생성할 때 사용자 지정 암호를 구성하면, 데이터가 저장되기 전에 AppVault가 즉시 사용자 지정 암호를 사용합니다. 암호를 변경할 필요가 없습니다.

AppVault에 이미 Kopia 또는 Restic 저장소가 있는 경우 새 암호를 추가해도 해당 저장소의 암호는 변경되지 않습니다. 이 섹션의 단계를 사용하지 마십시오. 대신 볼트 비밀번호를 변경하십시오의 단계를 따르십시오.

해당 `VAULT_PASSWORD` 값은 다음 요구 사항을 충족해야 합니다:

- 비어있으면 안 됩니다
- 크기는 64 KiB보다 작아야 합니다
- NUL 문자를 포함해서는 안 됩니다.
- 줄 바꿈을 포함해서는 안 됩니다
- 공백이나 기타 공백 문자로 시작하거나 끝나서는 안 됩니다

비밀번호에는 비 ASCII 문자를 포함한 다른 문자가 포함될 수 있습니다.

단계

1. AppVault와 동일한 네임스페이스에 Secret을 생성합니다. Secret에는 `VAULT_PASSWORD`라는 이름의 키가 포함되어야 합니다:

```
kubectl -n trident-protect create secret generic vault-password \
  --from-literal=VAULT_PASSWORD='<password>'
```

참고 AppVault에서 암호가 유효하지 않다고 보고하는 경우, `VAULT_PASSWORD` 값이 위의 요구 사항을 충족하는지 확인하십시오. 일반적인 원인으로는 암호 값이 비어 있거나, 끝에 줄 바꿈 문자가 있거나, 암호의 시작이나 끝에 공백이 있는 경우 등이 있습니다.

2. AppVault CR의 `spec.vaultPasswordSecretRef` 필드에서 Secret을 참조하십시오:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: my-vault
  namespace: trident-protect
spec:
  vaultPasswordSecretRef: vault-password
  providerType: <provider-type>
  providerConfig:
    <provider-configuration>
```

`spec.vaultPasswordSecretRef`이(가) 비어 있으면 AppVault는 안전하지 않은 기본 암호를 사용합니다.

결과

AppVault는 사용자 지정 암호를 사용하여 메타데이터 암호화 키를 보호합니다.

볼트 비밀번호를 변경하십시오

이미 데이터가 저장된 AppVault의 암호를 변경할 수 있습니다. 한 번 변경하면 메타데이터 암호화 키가 재암호화되고 AppVault의 모든 Kopia 리포지토리 및 모든 Restic 리포지토리가 새 암호로 이동됩니다.

이 작업에 대해

순환하는 동안 Trident Protect는 다음과 같은 작업을 수행합니다:

- AppVault에 대해 실행 중인 백업 및 복원 작업이 완료될 때까지 기다린 후 시작합니다.
- 새 비밀번호로 메타데이터 암호화 키(`metadata-key.json`)를 보호합니다. 아카이브 오브젝트는 다시 쓰지 않습니다.
- AppVault 접두사 아래에 있는 각 Kopia 저장소와 각 Restic 저장소를 찾아 비밀번호를 새 비밀번호로 변경합니다. 이미 새 비밀번호로 열리는 저장소는 변경되지 않습니다.
- 백업 또는 복원 작업에서 여전히 사용 중인 저장소는 건너뛰고 나중에 다시 시도합니다.
- AppVault를 참조하는 활성화된 일정을 일시 중지합니다. 각 일정에 대해 Trident Protect는 `spec.enabled`을(를) `false`(으)로 설정하고 다음 주석을 추가합니다:

```
protect.trident.netapp.io/vault-password-rotation-suspended:
<namespace>/<appvault-name>
```

순환이 완료되면 Trident Protect는 이 주석이 있는 일정만 다시 활성화합니다. 순환에 실패하면 중단된 지점부터 다시 시작합니다. 이미 새 암호를 사용하는 저장소는 다시 변경되지 않습니다.

시작하기 전에

다음 사항을 확인하십시오.

- 볼트 비밀번호 요구 사항을(를) 충족하는 새 비밀번호.
- 다른 AppVault는 암호를 변경하고 있지 않습니다. 여러 AppVaults가 동일한 버킷과 접두사를 공유하는 경우, 그중 하나만 암호를 변경할 수 있습니다. 버킷을 공유하는 AppVaults의 로테이션을 관리합니다을(를) 참조하십시오.

단계

1. 현재 Secret과 다른 이름으로 새 Secret을 생성하십시오:

```
kubectl -n trident-protect create secret generic vault-password-new \
  --from-literal=VAULT_PASSWORD='<new-password>'
```

중요함	순환이 완료되기 전에 현재 시크릿을 수정하거나 삭제하지 마십시오. Trident Protect는 모든 저장소가 새 암호를 사용할 때까지 기존 데이터를 열기 위해 현재 암호가 필요합니다. 현재 시크릿을 너무 일찍 삭제한 경우, 동일한 이름과 원래 `VAULT_PASSWORD` 값으로 다시 생성하십시오.

2. 회전을 활성화하고 AppVault가 새 Secret을 사용하도록 설정합니다. `my-vault`을(를) AppVault 이름으로 바꿉니다:

```
kubectl -n trident-protect patch appvault my-vault --type merge --patch
'
spec:
  enableRotation: true
  vaultPasswordSecretRef: vault-password-new
'
```

참고 | 비밀번호를 변경하기 전에 `spec.enableRotation`을(를) `true`(으)로 설정해야 합니다. 이 값이 `false`인 경우, AppVault는 순환이 비활성화되었다고 보고하고 비밀번호를 변경하지 않습니다.

3. 비밀번호 변경 모니터링의 단계를 따라 회전을 모니터링하십시오.
4. 교체가 완료되면 다른 곳에서 사용하지 않는 이전 Secret을 삭제합니다.

결과

메타데이터 암호화 키와 AppVault의 모든 Kopia 및 Restic 리포지토리는 새 암호로 보호되며, Trident Protect가 일시 중단한 일정이 다시 활성화됩니다.

비밀번호 변경 모니터링

비밀번호 변경 진행 상황을 확인하고 완료 여부를 확인할 수 있습니다.

단계

1. AppVault 보기:

```
kubectl -n trident-protect get appvault my-vault --output yaml
```

2. 출력에서 `status.vaultPasswordRotation` 섹션을 확인하십시오. 회전이 진행되는 동안에는 다음과 같이 표시됩니다:

```
status:
  vaultPasswordRotation:
    state: Rotating
    rotated: 3
    pending: 1
    message: <message>
```

이 표는 필드를 설명합니다:

필드에 입력합니다	설명
status.vaultPasswordRotation.state	현재 회전 상태.
status.vaultPasswordRotation.rotated	새 비밀번호를 사용하는 저장소 수입니다.

필드에 입력합니다	설명
status.vaultPasswordRotation.pending	백업 또는 복원 작업에서 사용 중이어서 건너뛴 저장소의 수입니다.
상태.vaultPasswordRotation.메시지	최신 진행 상황 또는 실패 메시지.

결과

회전은 다음 조건이 모두 충족될 때 완료됩니다:

- `status.vaultPasswordRotation`이(가) 존재하지 않습니다.
- `status.previousRepositoryPasswordSecretRef` 비어 있습니다.
- `status.repositoryPasswordSecretRef` 일치합니다 `spec.vaultPasswordSecretRef`.

다음 단계

회전이 완료되지 않으면 다음 사항을 확인하십시오:

- 회전이 시작되지 않음: `spec.enableRotation`이(가) `false`일 수 있습니다. 이를 `true`(으)로 설정한 다음, `spec.vaultPasswordSecretRef`을(를) 새 Secret으로 설정하십시오.
- 저장소가 대기 상태로 유지됨: 백업 또는 복원 작업에서 해당 저장소를 사용 중입니다. 작업이 완료될 때까지 기다리십시오. Trident Protect는 대기 중인 저장소에 대해 자동으로 재시도합니다.
- 다른 **AppVault**가 잠금을 소유하고 있어 회전이 시작되지 않습니다: 버킷을 공유하는 AppVaults의 로테이션을 관리합니다을(를) 참조하십시오.
- 메시지 필드에 오류가 보고되었습니다: 메시지에 설명된 문제를 해결하십시오. 로테이션은 중단된 지점부터 다시 시작됩니다.

dataMoverPasswordSecretRef에서 마이그레이션

기존 AppVault이 이전 `spec.dataMoverPasswordSecretRef` 필드를 사용하는 경우, 해당 필드를 `spec.vaultPasswordSecretRef`(으)로 이동할 수 있습니다.

이 작업에 대해

Trident Protect는 하위 호환성을 위해 `spec.dataMoverPasswordSecretRef`를 유지합니다. 새로운 AppVaults에는 `spec.vaultPasswordSecretRef`를 사용하십시오.

단계

1. `spec.dataMoverPasswordSecretRef` 및 해당 Secret을 그대로 유지하십시오.
2. `spec.vaultPasswordSecretRef`을(를) 사용하여 암호를 새 Secret으로 변경하십시오. 볼트 비밀번호를 변경하십시오의 단계를 따르십시오.
3. 순환 과정이 완료되면 AppVault에서 `spec.dataMoverPasswordSecretRef`을(를) 제거합니다.
4. 다른 곳에서 사용하지 않는 경우 기존 Secret을 삭제하십시오.

버킷을 공유하는 AppVaults의 로테이션을 관리합니다

여러 AppVaults가 동일한 버킷과 접두사를 사용하는 경우, 그중 하나만 암호를 변경할 수 있습니다.

이 작업에 대해

암호 순환에는 `password-rotation.lock`(이)라는 이름의 잠금 파일이 사용됩니다. 이 잠금에는 버킷 및 접두사에 대한 암호를 순환할 수 있는 AppVault의 UID가 기록됩니다. 두 번째 AppVault에 `spec.enableRotation: true`이(가) 있고 잠금에 다른 AppVault이(가) 지정되어 있으면 두 번째 AppVault은(는) 순환할 수 없습니다.

예 **1.** 단계

다른 **AppVault**로 로테이션 전송

1. 잠금을 소유한 AppVault에서 다음을 지웁니다 `spec.enableRotation`:

```
kubectl -n trident-protect patch appvault <current-owner> --type
merge --patch '
spec:
  enableRotation: false
'
```

2. 잠금이 해제될 때까지 기다리십시오.
3. 다음에 로테이션해야 할 AppVault에서 `spec.enableRotation`을(를) `true`(으)로 설정합니다:

```
kubectl -n trident-protect patch appvault <new-owner> --type merge
--patch '
spec:
  enableRotation: true
'
```

삭제된 **AppVault**가 남긴 잠금 제거

잠금을 소유한 AppVault가 더 이상 존재하지 않고 충돌 후와 같이 잠금이 제거되지 않은 경우 다음 단계를 따르십시오.

1. 버킷의 AppVault 접두사에서 `password-rotation.lock` 파일을 삭제하십시오.
2. 순환해야 하는 AppVault에서 `spec.enableRotation`을(를) `true`(으)로 설정합니다:

```
kubectl -n trident-protect patch appvault <appvault-name> --type
merge --patch '
spec:
  enableRotation: true
'
```

관련 정보

- "Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다."
- "Trident Protect 설치 사용자 정의"

- "애플리케이션 보호"
- "Trident Protect를 사용하여 애플리케이션 복원"

Trident Protect를 사용하여 관리를 위한 애플리케이션 정의

Trident Protect로 관리하려는 애플리케이션을 정의하려면 애플리케이션 CR과 관련 AppVault CR을 생성하면 됩니다.

AppVault CR을 작성합니다

애플리케이션에서 데이터 보호 작업을 수행할 때 사용할 AppVault CR을 만들어야 하며, AppVault CR은 Trident Protect가 설치된 클러스터에 있어야 합니다. AppVault CR은 사용자 환경에 따라 다릅니다. AppVault CR의 예는 다음을 참조하세요."AppVault 사용자 지정 리소스."

응용 프로그램을 정의합니다

Trident Protect로 관리하려는 각 애플리케이션을 정의해야 합니다. 수동으로 애플리케이션 CR을 생성하거나 Trident Protect CLI를 사용하여 관리할 애플리케이션을 정의할 수 있습니다.

참고	전체 네임스페이스가 아닌 이름으로 특정 KubeVirt 가상 머신(VM)을 보호하는 VM 범위 애플리케이션을 정의하려면 애플리케이션 CR의 `spec.includedVirtualMachines` 필드 또는 CLI의 `--virtual-machines` 플래그를 사용하십시오. 자세한 내용, 필드 참조 및 YAML 예제는 "KubeVirt 가상 머신을 Trident Protect로 보호합니다"을 참조하십시오.

CR을 사용하여 응용 프로그램을 추가합니다

단계

1. 대상 응용 프로그램 CR 파일을 만듭니다.
   a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `maria-app.yaml`).
   b. 다음 특성을 구성합니다.
      - **metadata.name**: (*required*) 응용 프로그램 사용자 정의 리소스의 이름입니다. 보호 작업에 필요한 다른 CR 파일이 이 값을 참조하기 때문에 선택하는 이름을 확인합니다.
      - **spec.includedNamespaces**: (*required*) 네임스페이스 및 레이블 선택기를 사용하여 응용 프로그램에서 사용하는 네임스페이스와 리소스를 지정합니다. 응용 프로그램 네임스페이스는 이 목록의 일부여야 합니다. 레이블 선택기는 선택 사항이며 지정된 각 네임스페이스 내에서 리소스를 필터링하는 데 사용할 수 있습니다.
      - **spec.includedClusterScopedResources**: (*Optional*) 이 속성을 사용하여 응용 프로그램 정의에 포함될 클러스터 범위 리소스를 지정합니다. 이 속성을 사용하면 해당 그룹, 버전, 종류 및 레이블을 기반으로 이러한 리소스를 선택할 수 있습니다.
        - **groupVersionKind**: (*required*) 클러스터 범위 리소스의 API 그룹, 버전 및 종류를 지정합니다.
        - **labelSelector**: (*Optional*) 레이블을 기반으로 클러스터 범위 리소스를 필터링합니다.
      - **spec.volumeGroups**: (선택 사항) 이 속성을 사용하여 관련 볼륨을 크래시 일관성 볼륨 그룹 스냅샷으로 함께 스냅샷할 수 있습니다. 자세한 내용은 "Trident Protect에서 볼륨 그룹 스냅샷 사용"을(를) 참조하십시오.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (선택 사항) 이 주석은

KubeVirt 환경과 같이 스냅샷 전에 파일 시스템이 정지되는 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 스냅샷 중에 이 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정합니다. true로 설정하면 애플리케이션은 글로벌 설정을 무시하고 스냅샷 중에 파일 시스템에 쓸 수 있습니다. false로 설정하면 애플리케이션은 글로벌 설정을 무시하고 스냅샷을 찍는 동안 파일 시스템이 동결됩니다. 지정되었지만 애플리케이션 정의에 가상 머신이 없는 경우 주석은 무시됩니다. 지정하지 않으면 응용 프로그램은 다음을 따릅니다."글로벌 Trident 프로텍트 동결 설정" .

응용 프로그램이 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) 필요한 경우 동일한 CR에 리소스 필터링을 추가하여 특정 리소스를 포함하거나 제외할 수

있습니다.

◦ 일반 필터 예:

- **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) `Include` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 `Exclude` 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
  - **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
  - **resourceMatchers[].group**:(*Optional*) 필터링할 리소스의 그룹입니다.
  - * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
  - **resourceMatchers [ ].version**: (*Optional*) 필터링할 리소스의 버전입니다.
  - **resourceMatchers[].names**:(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
  - *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - **resourceMatchers [ ].labelSelectors**: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "Kubernetes 문서" 예를 들면 다음과 `"trident.netapp.io/os=linux"`같습니다.

참고	둘 다 `resourceFilter` 그리고 `labelSelector` 사용됩니다, `resourceFilter` 먼저 실행한 다음 `labelSelector` 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"]
```

◦ **PVC 전용 필터 예:**

PVC 전용 애플리케이션을 정의하려면 리소스 필터에 `PersistentVolume` 및 `VolumeSnapshotClass`도 포함해야 합니다. 스냅샷 및 백업 작업은 `PersistentVolume`(각 PVC에 바인딩된 클러스터 범위 볼륨) 및 `VolumeSnapshotClass`(스냅샷 드라이버)에 따라 달라지며 이들이 없으면 실패합니다. 예를 들면 다음과 같습니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. 환경에 맞게 응용 프로그램 CR을 만든 후 CR을 적용합니다. 예를 들면 다음과 같습니다.

```
kubectl apply -f maria-app.yaml
```

단계

1. 다음 예제 중 하나를 사용하여 응용 프로그램 정의를 만들고 적용합니다. 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다. 예제에 표시된 인수를 사용하여 쉼표로 구분된 목록을 사용하여 응용 프로그램 정의에 네임스페이스와 리소스를 포함할 수 있습니다.

스냅샷 중에 애플리케이션이 파일 시스템에 쓸 수 있는지 여부를 지정하기 위해 앱을 생성할 때 주석을 사용할 수 있습니다. 이는 스냅샷 전에 파일 시스템이 정지되는 KubeVirt 환경과 같은 가상 머신에서 정의된 애플리케이션에만 적용됩니다. 주석을 다음과 같이 설정하면 `true` , 애플리케이션은 글로벌 설정을 무시하고 스냅샷 동안 파일 시스템에 쓸 수 있습니다. 이것을 설정하면 `false` , 애플리케이션은 글로벌 설정을 무시하고 스냅샷을 찍는 동안 파일 시스템이 동결됩니다. 주석을 사용하지만 애플리케이션 정의에 가상 머신이 없는 경우 주석은 무시됩니다. 주석을 사용하지 않으면 응용 프로그램은 다음을 따릅니다."글로벌 Trident 프로텍트 동결 설정" .

+ CLI를 사용하여 애플리케이션을 생성할 때 주석을 지정하려면 `--annotation` 플래그를 사용할 수 있습니다.

+ * 애플리케이션을 생성하고 파일 시스템 고정 동작에 대한 전역 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 동결 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 예시와 같이 그룹, 종류, 버전, 레이블, 이름, 네임스페이스 등의 `resourceSelectionCriteria`을(를) 기준으로 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"]}]'
  ```

  + * PVC 전용 애플리케이션을 정의하려면 `PersistentVolume` 및 `VolumeSnapshotClass``도 리소스 필터에 포함해야 합니다. 스냅샷 및 백업 작업은 `PersistentVolume`(각 PVC에 바인딩된 클러스터 범위 볼륨) 및 `VolumeSnapshotClass`(스냅샷 드라이버)에 따라 달라지며, 이들이 없으면 실패합니다. 예를 들면 다음과 같습니다.

  ```
  tridentctl-protect create app my-pvc-app --namespaces <my-app-
  namespace> --resource-filter-include
  '[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"Persisten
  tVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1
  "}]' -n <my-app-namespace>
  ```

Trident Protect를 사용하여 애플리케이션 보호

자동화된 보호 정책이나 임시 기반으로 스냅샷과 백업을 수행하여 Trident Protect가 관리하는 모든 앱을 보호할 수 있습니다.

참고

- 데이터 보호 작업 중에 파일 시스템을 동결 및 동결 해제하도록 Trident Protect를 구성할 수 있습니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".
- 애플리케이션의 관련 볼륨을 동일한 시점에 스냅샷하려면 애플리케이션에서 볼륨 그룹을 정의하십시오. "Trident Protect에서 볼륨 그룹 스냅샷 사용"을(를) 참조하십시오.

필요 시 스냅샷을 생성합니다

언제든지 주문형 스냅샷을 생성할 수 있습니다.

참고

클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 스냅샷을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.applicationRef *: 스냅샷을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - * spec.appVaultRef *: (*required*) 스냅샷 내용(메타데이터)을 저장할 AppVault의 이름입니다.
   - **spec.reclaimPolicy**: (*Optional*) 스냅샷 CR이 삭제될 때 스냅샷의 AppArchive에 발생하는 동작을 정의합니다. 즉, 로 설정된 경우에도 `Retain` 스냅샷이 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. 파일을 올바른 값으로 채운 후 `trident-protect-snapshot-cr.yaml` CR:

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

CLI를 사용하여 스냅샷을 생성합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 생성합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

필요 시 백업을 생성합니다

언제든지 앱을 백업할 수 있습니다.

참고	클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 백업 작업에 충분한지 확인합니다. 백업 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS IAM 설명서" 참조하십시오.

CR을 사용하여 백업을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
    - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
    - * spec.applicationRef *:(*required*) 백업할 응용 프로그램의 Kubernetes 이름입니다.
    - * spec.appVaultRef *: (*required*) 백업 내용을 저장할 AppVault의 이름입니다.
    - * spec.datamover *: (*Optional*) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
        - `Restic`
        - `Kopia` (기본값)
    - **spec.reclaimPolicy**: (*Optional*) 클레임에서 해제될 때 백업에 어떤 일이 발생하는지를 정의합니다. 가능한 값:
        - `Delete`
        - `Retain` (기본값)
    - **spec.snapshotRef**: (선택 사항): 백업 소스로 사용할 스냅샷의 이름입니다. 제공하지 않으면 임시

스냅샷이 생성되고 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. 파일을 올바른 값으로 채운 후 `trident-protect-backup-cr.yaml` CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

CLI를 사용하여 백업을 생성합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 백업을 만듭니다. 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>
```

필요에 따라 플래그를 사용하여 백업이 비증분 백업이어야 하는지 여부를 지정할 수 `--full-backup` 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 이 플래그를 사용하면 백업이 비증분 백업이 됩니다. 전체 백업을 주기적으로 수행한 다음 전체 백업 사이에 증분 백업을 수행하여 복원과 관련된 위험을 최소화하는 것이 좋습니다.

지원되는 백업 주석

다음 표에서는 백업 CR을 만들 때 사용할 수 있는 주석을 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/전체 백업	문자열	백업이 증분적이지 않아야 하는지 여부를 지정합니다. 로 설정 `true` 비증분 백업을 생성합니다. 복원과 관련된 위험을 최소화하려면 주기적으로 전체 백업을 수행하고 전체 백업 사이에 증분 백업을 수행하는 것이 가장 좋습니다.	"거짓"

주석	유형	설명	기본값
protect.trident.netapp.io/스냅샷 완료 시간 초과	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/볼륨 스냅샷 사용 준비 시간 초과	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30m"
protect.trident.netapp.io/볼륨 스냅샷 생성 시간 초과	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 도달할 때까지 기다리는 최대 시간(초) `Bound` 작업이 실패하기 전 단계입니다.	"1200"(20분)

데이터 보호 스케줄을 생성합니다

보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 매시간, 매일, 매주, 매월 스냅샷과 백업을 만들도록 선택할 수 있으며, 보관할 복사본 수를 지정할 수 있습니다. full-backup-rule 주석을 사용하여 비증분 전체 백업을 예약할 수 있습니다. 기본적으로 모든 백업은 증분 백업입니다. 주기적으로 전체 백업을 수행하고 그 사이에 증분 백업을 수행하면 복원과 관련된 위험을 줄이는 데 도움이 됩니다.

참고

- 스냅샷에 대한 일정은 다음을 설정하여 생성할 수 있습니다. `backupRetention` 0으로 그리고 `snapshotRetention` 0보다 큰 값으로. 환경 `snapshotRetention` 0으로 설정하면 예약된 백업은 여전히 스냅샷을 생성하지만, 이는 임시적이며 백업이 완료되면 즉시 삭제됩니다.
- 클러스터 범위 리소스는 애플리케이션 정의에 명시적으로 참조되거나 애플리케이션 네임스페이스에 대한 참조가 있는 경우 백업, 스냅샷 또는 클론에 포함됩니다.

CR을 사용하여 일정을 생성합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-schedule-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
    - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
    - * spec.datamover *: (*Optional*) 백업 작업에 사용할 백업 도구를 나타내는 문자열입니다. 가능한 값(대소문자 구분):
        - `Restic`
        - `Kopia` (기본값)
    - * spec.applicationRef *: 백업할 응용 프로그램의 Kubernetes 이름입니다.
    - * spec.appVaultRef *: (*required*) 백업 내용을 저장할 AppVault의 이름입니다.
    - **spec.backupRetention**: (필수) 보관할 백업 수입니다. 0은 백업을 생성하지 않음을 나타냅니다(스냅샷만 생성).
    - **spec.backupReclaimPolicy**: (선택 사항) 보존 기간 동안 백업 CR이 삭제된 경우 백업에 어떤 일이 발생하는지 결정합니다. 보관 기간이 지나면 백업은 항상 삭제됩니다. 가능한 값(대소문자 구분):
        - `Retain` (기본값)

- `Delete`

- **spec.snapshotRetention**: (필수) 보관할 스냅샷 수입니다. 0은 스냅샷을 생성하지 않음을 나타냅니다.
- **spec.snapshotReclaimPolicy**: (선택 사항) 스냅샷 CR이 보존 기간 동안 삭제된 경우 스냅샷에 어떤 일이 발생하는지 결정합니다. 보존 기간이 지나면 스냅샷은 항상 삭제됩니다. 가능한 값(대소문자 구분):
  - `Retain`
  - `Delete`(기본)
- **spec.granularity**: 일정이 실행되는 빈도. 가능한 값과 필수 관련 필드:
  - `Hourly`(지정해야 함) `spec.minute`)
  - `Daily`(지정해야 함) `spec.minute` 그리고 `spec.hour`)
  - `Weekly`(지정해야 함) `spec.minute, spec.hour` , 그리고 `spec.dayOfWeek`)
  - `Monthly`(지정해야 함) `spec.minute, spec.hour` , 그리고 `spec.dayOfMonth`)
  - `Custom`
- **spec.dayOfMonth**: (선택 사항) 일정을 실행해야 하는 날짜(1~31)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. `Monthly` . 값은 문자열로 제공되어야 합니다.
- **spec.dayOfWeek**: (선택 사항) 일정을 실행해야 하는 요일(0~7). 0 또는 7의 값은 일요일을 나타냅니다. 세분성이 설정된 경우 이 필드가 필요합니다. `Weekly` . 값은 문자열로 제공되어야 합니다.
- **spec.hour**: (선택 사항) 일정을 실행해야 하는 시간(0~23)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. `Daily` , `Weekly` , 또는 `Monthly` . 값은 문자열로 제공되어야 합니다.
- **spec.minute**: (선택 사항) 일정을 실행해야 하는 분(0~59)입니다. 세분성이 설정된 경우 이 필드가 필요합니다. `Hourly` , `Daily` , `Weekly` , 또는 `Monthly` . 값은 문자열로 제공되어야 합니다.
- **spec.runImmediately**: (선택 사항) `true`로 설정하면 스케줄 생성 시 일회성으로 즉시 기준선 실행(보존 설정에 따른 백업 및/또는 스냅샷)을 트리거합니다. 기본값은 `false`입니다. 이 설정은 이후 반복 실행에는 영향을 미치지 않습니다.

  백업 및 스냅샷 일정에 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 일정에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

즉시 실행이 포함된 스케줄에 대한 YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. 파일을 올바른 값으로 채운 후 `trident-protect-schedule-cr.yaml` CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

CLI를 사용하여 일정을 생성합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 보호 스케줄을 생성합니다. 예를 들면 다음과 같습니다.

참고	을 사용하여 `tridentctl-protect create schedule --help` 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: 다음을 사용하세요 `--full-backup-rule` 증분 백업이 아닌 전체 백업을 예약하려면 플래그를 지정하세요. 이 플래그는 다음 경우에만 작동합니다. `--granularity Daily`. 가능한 값:
  - `Always`매일 전체 백업을 생성하세요.
  - 특정 요일 지정: 쉼표로 구분하여 하나 이상의 요일을 지정하세요(예: `"Monday,Thursday"`). 유효한 값: 월요일, 화요일, 수요일, 목요일, 금요일, 토요일, 일요일.

참고	그만큼 `--full-backup-rule` 해당 플래그는 시간별, 주별 또는 월별 세분화에서는 작동하지 않습니다.

- 즉시 기준선 보호: `--run-immediately true`을 사용하여 첫 번째 예약 실행 시간을 기다리지 않고 예약이 생성될 때 즉시 초기 백업 또는 스냅샷을 생성합니다. 기본값은 `false`입니다.
- 스냅샷 전용 일정: 설정 `--backup-retention 0` 그리고 0보다 큰 값을 지정하십시오. `--snapshot-retention`.

지원되는 일정 주석

다음 표에서는 일정 CR을 만들 때 사용할 수 있는 주석을 설명합니다.

주석	유형	설명	기본값
protect.trident.netapp.io/전체 백업 규칙	문자열	전체 백업 예약 규칙을 지정합니다. 다음과 같이 설정할 수 있습니다. `Always` 지속적인 전체 백업을 원하시거나 필요에 따라 맞춤 설정할 수 있습니다. 예를 들어, 일별 세분성을 선택하면 전체 백업이 수행될 요일을 지정할 수 있습니다(예: `"Monday,Thursday"`). 유효한 요일 값은 월요일, 화요일, 수요일, 목요일, 금요일, 토요일, 일요일입니다. 참고로 이 주석은 특정 조건을 충족하는 일정에만 사용할 수 있습니다. `granularity` 설정됨 `Daily`.	설정되지 않음 (모든 백업은 증분 백업임)
protect.trident.netapp.io/스냅샷 완료 시간 초과	문자열	전체 스냅샷 작업이 완료되는 데 허용되는 최대 시간입니다.	"60m"
protect.trident.netapp.io/볼륨 스냅샷 사용 준비 시간 초과	문자열	볼륨 스냅샷이 사용 가능한 상태에 도달하는 데 허용되는 최대 시간입니다.	"30m"
protect.trident.netapp.io/볼륨 스냅샷 생성 시간 초과	문자열	볼륨 스냅샷을 생성하는 데 허용되는 최대 시간입니다.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	문자열	새로 생성된 PersistentVolumeClaims(PVC)가 도달할 때까지 기다리는 최대 시간(초) `Bound` 작업이 실패하기 전 단계입니다.	"1200"(20분)

스냅샷을 삭제합니다

더 이상 필요하지 않은 예약된 스냅샷 또는 주문형 스냅샷을 삭제합니다.

단계

1. 스냅샷과 연결된 스냅샷 CR을 제거합니다.

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

백업을 삭제합니다

더 이상 필요하지 않은 예약된 백업 또는 필요 시 백업을 삭제합니다.

참고

회수 정책이 설정되어 있는지 확인하십시오. `Delete` 개체 저장소에서 모든 백업 데이터를 제거합니다. 정책의 기본 설정은 다음과 같습니다. `Retain` 실수로 인한 데이터 손실을 방지하기 위해. 정책이 변경되지 않으면 `Delete` 백업 데이터는 개체 스토리지에 남아 있으며 수동으로 삭제해야 합니다.

단계

1. 백업과 연결된 백업 CR을 제거합니다.

```
kubectl delete backup <backup_name> -n my-app-namespace
```

백업 작업의 상태를 확인합니다

명령줄을 사용하여 진행 중이거나, 완료되었거나, 실패한 백업 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 백업 작업의 상태를 검색하여 대괄호의 값을 사용자 환경의 정보로 바꿉니다.

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Azure-NetApp-files(ANF) 작업을 위한 백업 및 복원이 가능합니다

Trident Protect를 설치한 경우 azure-netapp-files 스토리지 클래스를 사용하고 Trident 24.06 이전에 생성된 스토리지 백엔드에 대해 공간 효율적인 백업 및 복원 기능을 활성화할 수 있습니다. 이 기능은 NFSv4 볼륨에서 작동하며 용량 풀에서 추가 공간을 소모하지 않습니다.

시작하기 전에

다음을 확인합니다.

- Trident Protect를 설치했습니다.
- Trident Protect에서 애플리케이션을 정의했습니다. 이 절차를 완료하기 전까지 이 애플리케이션의 보호 기능은 제한됩니다.
- `azure-netapp-files`스토리지 백엔드의 기본 스토리지 클래스로 선택했습니다.

▼ 구성 단계를 위해 확장합니다

1. Trident 24.10으로 업그레이드하기 전에 ANF 볼륨을 생성한 경우 Trident에서 다음을 수행하십시오.

   a. Azure-NetApp-files 기반이며 애플리케이션과 연결된 각 PV에 대해 스냅샷 디렉토리를 활성화합니다.

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. 연결된 각 PV에 대해 스냅샷 디렉토리가 활성화되었는지 확인합니다.

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   응답:

   ```
   snapshotDirectory: "true"
   ```

   +
   스냅샷 디렉토리가 활성화되지 않은 경우 Trident Protect는 정기적인 백업 기능을 선택합니다. 이 기능은 백업 프로세스 중에 용량 풀의 공간을 일시적으로 사용합니다. 이 경우 백업되는 볼륨 크기의 임시 볼륨을 생성할 수 있는 충분한 공간이 용량 풀에 있는지 확인하세요.

결과

Trident Protect를 사용하여 애플리케이션을 백업하고 복원할 준비가 되었습니다. 각 PVC는 다른 애플리케이션에서 백업 및 복원을 위해 사용할 수도 있습니다.

Trident Protect에서 볼륨 그룹 스냅샷 사용

볼륨 그룹 스냅샷을 사용하면 관련된 영구 볼륨 클레임(PVC)을 동일한 시점에 캡처할 수 있습니다. 이는 여러 디스크를 사용하는 가상 머신(VM)을 포함하여 둘 이상의 볼륨에 쓰는 애플리케이션에 유용합니다.

이 작업에 대해

`spec.volumeGroups` 필드와 `Application` 사용자 정의 리소스(CR)에서 볼륨 그룹을 정의합니다. 애플리케이션의 스냅샷을 생성할 때 Trident Protect는 ONTAP 일관성 그룹을 사용하여 볼륨 그룹의 모든 PVC를 동시에 스냅샷합니다. 볼륨 그룹의 각 네임스페이스에 대해 하나의 일관성 그룹이 생성됩니다. 그룹의 각 볼륨에는 자체 스냅샷도 생성되므로 볼륨을 개별적으로 복원할 수 있습니다.

Trident Protect는 볼륨 그룹에 속하지 않은 PVC에 대해 개별 스냅샷을 생성합니다. 이러한 스냅샷은 순차적으로 생성되므로 동일한 시점을 반영한다고 보장할 수 없습니다.

볼륨 그룹 스냅샷은 요청 시 스냅샷, 예약된 스냅샷, 그리고 Trident Protect가 백업의 일부로 생성하는 스냅샷에 적용됩니다. 기존 스냅샷을 백업하는 경우 Trident Protect는 해당 스냅샷을 사용하며 새 그룹 스냅샷을 생성하지 않습니다. 복제 또한 새로운 그룹 스냅샷을 생성하지 않습니다. 그룹 스냅샷이 이미 생성한 볼륨별 스냅샷을 사용합니다.

Trident Protect는 다음 조건을 모두 충족하는 경우에만 동일한 시점에 볼륨을 캡처합니다:

- 동일한 볼륨 그룹에 있습니다.
- 동일한 네임스페이스에 있습니다.
- 이들은 동일한 스토리지 가상 머신(SVM)에 있습니다.
- 해당 볼륨 그룹에는 해당 네임스페이스에 최대 80개의 볼륨이 있습니다.

요구 사항이 충족되지 않는 등의 이유로 Trident Protect가 그룹 스냅샷을 생성할 수 없는 경우, `status.warnings`에 경고를 추가하고 해당 그룹의 PVC에 대한 개별 스냅샷을 생성합니다. 이러한 스냅샷은 순차적으로 생성되므로 동일한 시점을 반영하지 않을 수 있습니다. 애플리케이션 스냅샷은 여전히 완료됩니다.

다음 표는 일반적인 상황에서 발생하는 일을 설명합니다.

상황	결과
볼륨 그룹은 둘 이상의 네임스페이스에서 PVC를 선택합니다.	그룹 스냅샷은 하나의 네임스페이스에 속하므로 Trident Protect는 각 네임스페이스에 별도의 그룹 스냅샷을 생성합니다. 이러한 그룹 스냅샷은 동시에 생성되지 않습니다. 하나가 성공하는 동안 다른 하나는 아직 실행 중이거나, 실패하거나, 개별 스냅샷으로 대체될 수 있습니다. 예를 들어, `app: payments-db`에 레이블이 지정된 PVC는 `payments-a`와 `payments-b`에서 두 개의 별도 그룹 스냅샷이 됩니다. 서로 다른 네임스페이스의 그룹 스냅샷을 하나의 복원 지점으로 취급하지 마십시오.

상황	결과
네임스페이스의 PVC가 둘 이상의 SVM에 있습니다.	Trident Protect는 SVM 또는 Trident 백엔드별로 볼륨을 분할하지 않습니다. ONTAP 일관성 그룹은 둘 이상의 SVM에 속한 볼륨을 포함할 수 없으므로, 동일한 SVM을 공유하는 볼륨을 포함하여 전체 네임스페이스에 대한 그룹 스냅샷이 실패합니다. Trident Protect는 경고를 추가하고, 그룹 스냅샷을 제거하며, 각 볼륨의 개별 스냅샷을 생성합니다. 볼륨을 함께 캡처하려면 동일한 SVM에 배치하십시오.
네임스페이스에 일치하는 PVC가 80개 이상 있습니다.	Trident Protect는 해당 네임스페이스만 건너뛰고 해당 PVC에 대한 개별 스냅샷을 생성합니다. 동일한 볼륨 그룹에 있는 다른 네임스페이스에는 여전히 그룹 스냅샷이 생성됩니다.
PVC는 둘 이상의 `volumeGroups` 항목과 일치합니다	Trident Protect는 PVC를 두 볼륨 그룹 모두에 추가하며, 그룹 스냅샷 중 하나가 실패할 수 있습니다. 실패한 그룹 스냅샷은 개별 스냅샷으로 대체됩니다. 각 PVC가 하나의 항목에만 일치하는지 확인하십시오.
볼륨 그룹에는 볼륨 그룹 스냅샷을 지원하지 않는 드라이버 또는 프로토콜을 사용하는 PVC가 포함됩니다	그룹 스냅샷이 실패하고, Trident Protect는 해당 PVC의 개별 스냅샷을 생성합니다.

시작하기 전에

- PVC는 볼륨 그룹 스냅샷을 지원하는 스토리지 드라이버와 프로토콜을 사용해야 합니다. 지원되는 드라이버 및 프로토콜은 "Trident 볼륨 그룹 스냅샷 작업"을(를) 참조하십시오.
- 클러스터는 Kubernetes 1.36 이상 버전을 실행해야 합니다.
- Kubernetes 스냅샷 컨트롤러는 버전 8.5.0 이상이어야 하며, `CSIVolumeGroupSnapshot` 기능 게이트가 활성화되어 있어야 합니다.
- `v1`, `VolumeGroupSnapshot`, `VolumeGroupSnapshotClass`, `VolumeGroupSnapshotContent`에 대한 CRD가 설치되어 있어야 합니다.
- 온라인 백엔드에서는 Trident 26.10 이상을 사용해야 합니다. `CSIVolumeGroupSnapshot` 기능 게이트는 Trident `csi-snapshotter` 컨테이너에서도 활성화되어 있어야 합니다.
- `VolumeGroupSnapshotClass`를 사용하는 A `csi.trident.netapp.io` 드라이버를 정의해야 합니다. 클래스가 하나만 정의된 경우 Trident Protect가 해당 클래스를 사용합니다. 여러 클래스가 정의된 경우 Trident Protect가 사용해야 하는 클래스에 `snapshot.storage.kubernetes.io/is-default-class: "true"`로 주석을 답니다. 그렇지 않으면 `Trident Protect`는 알파벳 순서로 첫 번째 클래스를 사용합니다. `Application` 또는 `Snapshot` CR에서는 클래스를 선택할 수 없습니다. 자세한 내용은 "볼륨 그룹 스냅샷 클래스"을 참조하십시오.
- Trident 백엔드의 ONTAP 계정은 일관성 그룹을 생성, 가져오기, 삭제하고 스냅샷을 생성할 수 있는 권한이 있어야 합니다. "애플리케이션 일관성 그룹 관리" 및 "애플리케이션 일관성 그룹 스냅샷 관리"을(를) 참조하십시오. 기본 SVM 관리자 역할 (`vsadmin`)에는 이러한 권한이 포함되어 있지 않습니다.

볼륨 그룹을 정의합니다

Trident Protect는 볼륨 그룹에 속하는 볼륨에 대해서만 그룹 스냅샷을 생성합니다. `spec.volumeGroups`을 `Application` CR에 추가하여 그룹을 정의합니다. 각 항목은 별도의 그룹입니다. 레이블, VM 또는 둘 다를 기준으로 볼륨을 선택합니다.

이 작업에 대해

- 볼륨 그룹은 `Application` CR에서만 정의할 수 있습니다. `tridentctl-protect` CLI에는 볼륨 그룹 옵션이 없습니다.

- Trident Protect는 바인딩된 PVC만 볼륨 그룹에 포함하고 바인딩되지 않은 PVC에 대해서는 개별 스냅샷을 생성합니다. 만약 항목에서 바인딩된 PVC를 선택하지 않으면, Trident Protect는 해당 항목에 대한 그룹 스냅샷을 생성하지 않습니다.
- 각 PVC가 하나의 항목에만 일치하는지 확인하십시오.

시작하기 전에

보호할 애플리케이션을 정의하십시오. "Trident Protect를 사용하여 관리를 위한 애플리케이션 정의"을(를) 참조하십시오.

단계

1. 레이블로 볼륨을 선택하는 경우 그룹화할 PVC에 레이블을 지정하십시오. 어떤 레이블 키와 값이든 사용할 수 있습니다. 이 페이지의 예에서는 `volume-group=my-volume-group`를 사용합니다.

```
kubectl label pvc <pvc_name_1> <pvc_name_2> volume-group=my-volume-group
-n <application_namespace>
```

2. 기존 `Application` CR 파일에 `spec.volumeGroups`을(를) 추가합니다. 각 항목에는 `labelSelector`, `vmSelector` 또는 둘 다 포함되어야 합니다. 항목에 두 필드가 모두 포함된 경우, 볼륨 그룹에는 두 필드 중 하나가 선택하는 모든 볼륨이 포함됩니다.

   - **spec.volumeGroups[].labelSelector**: 애플리케이션에서 PVC를 선택하는 Kubernetes 레이블 선택기(`matchLabels` 또는 `matchExpressions`)입니다. 빈 선택기는 유효하지 않습니다.
   - **spec.volumeGroups[].vmSelector**: KubeVirt VM 목록이며, 각 VM에는 `namespace` 및 `name`가 있습니다. Trident Protect에는 각 VM이 마운트하는 PVC가 포함됩니다.

     다음 예는 두 개의 볼륨 그룹을 정의합니다. 첫 번째 그룹에는 `volume-group: my-volume-group` 레이블이 지정된 PVC가 포함됩니다. 두 번째 그룹에는 `my-vm-1` 및 `my-vm-2`이(가) 마운트하는 PVC가 포함됩니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-namespace
spec:
  includedNamespaces:
    - namespace: my-namespace
  volumeGroups:
    - labelSelector:
        matchLabels:
          volume-group: my-volume-group
    - vmSelector:
        - namespace: my-namespace
          name: my-vm-1
        - namespace: my-namespace
```

```
            name: my-vm-2
```

+

참고 | `Application` CR에 이미 포함된 다른 필드(예: `spec.resourceFilter` 등)는 그대로 유지하십시오. 적용하는 파일에서 필드를 생략하면 `kubectl apply`이(가) CR에서 해당 필드를 삭제합니다.

3. 업데이트된 CR을 적용하십시오:

```
kubectl apply -f <application_cr_file>.yaml
```

결과

`Application`CR에는 볼륨 그룹이 포함되어 있습니다. 애플리케이션의 다음 스냅샷에서 해당 볼륨 그룹이 사용됩니다.

스냅샷을 생성합니다

애플리케이션에 대한 온디맨드 스냅샷, 온디맨드 백업 또는 데이터 보호 일정을 생성합니다. 필드를 추가할 필요는 없습니다. Trident Protect는 `Application` CR에서 정의한 볼륨 그룹을 사용합니다. 지침은 "Trident Protect를 사용하여 애플리케이션 보호"을(를) 참조하십시오.

참고 | 기본적으로 Trident Protect는 VM 파일 시스템을 동결하기 전에 그룹 스냅샷을 생성합니다. 볼륨 그룹의 볼륨은 동결 없이 캡처되므로 VM이 아직 디스크에 기록하지 않은 데이터는 포함되지 않습니다. 볼륨 그룹에 속하지 않은 디스크는 Trident Protect가 VM을 동결한 후 개별 스냅샷을 생성할 때 나중에 캡처됩니다. VM의 모든 디스크를 동시에 캡처하려면 모든 디스크를 볼륨 그룹에 포함하십시오. VM 파일 시스템 동결에 대한 자세한 내용은 "KubeVirt VM으로 데이터 보호"을(를) 참조하십시오.

결과를 검토하십시오

스냅샷 생성이 완료되면 상태를 확인하여 Trident Protect가 그룹 스냅샷을 생성했는지 확인하십시오.

참고 | 애플리케이션 스냅샷이 완료되었다고 해서 해당 그룹 스냅샷이 성공한 것은 아닙니다. 모든 그룹 스냅샷이 건너뛰어지거나 실패하더라도 `VolumeGroupSnapshotsCreated` 조건은 `true`(이) 되며, Trident Protect는 Kubernetes 이벤트나 메트릭을 생성하지 않습니다. 이를 감지하려면 `status.warnings`와(과) `skipped` 및 `failed` 필드를 모니터링하십시오.

단계

1. 스냅샷의 이름을 가져옵니다. 애플리케이션의 스냅샷 목록을 보려면 다음 명령을 실행하십시오.

```
kubectl get snapshots -n <application_namespace>
```

2. 스냅샷 상태를 확인하십시오:

```
kubectl get snapshots <snapshot_name> -n <application_namespace> -o yaml
```

결과

`status.volumeGroupSnapshots` 필드에는 그룹 스냅샷마다 하나의 항목이 있습니다. Trident Protect는 볼륨 그룹의 각 네임스페이스에 대해 별도의 그룹 스냅샷을 생성합니다. 다음 예는 축약된 것입니다:

```
status:
  volumeGroupSnapshots:
    - name: vgs-<snapshot_uid_prefix>-<group_index>-<namespace>
      namespace: <namespace>
      ready: true
      memberPVCs:
        - pvcName: <pvc_name_1>
          pvcNamespace: <namespace>
        - pvcName: <pvc_name_2>
          pvcNamespace: <namespace>
```

그룹 스냅샷 이름에는 Snapshot UID의 처음 8자, spec.volumeGroups 항목의 인덱스(`0`부터 시작), 그리고 12자로 잘린 네임스페이스가 포함됩니다.

- `ready: true`은(는) 그룹 스냅샷이 성공했음을 의미합니다. 그룹을 구성하는 스냅샷도 `status.volumeSnapshots` 아래에 나열됩니다. 이는 그룹의 멤버 스냅샷이며, 동일한 볼륨의 추가 스냅샷이 아닙니다.
- `skipped: true`은(는) 볼륨 그룹이 해당 네임스페이스에 80개 이상의 PVC를 가지고 있기 때문에 Trident Protect가 그룹 스냅샷 생성을 시도하지 않았음을 의미합니다. 대신 Trident Protect는 그룹 볼륨의 개별 스냅샷을 생성하고 `skipReason` 텍스트를 `status.warnings`에 추가합니다.
- `failed: true` 는 Trident Protect가 그룹 스냅샷 생성을 시도했지만 실패했음을 의미합니다. Trident Protect는 그룹 볼륨의 개별 스냅샷을 생성하고 `error` 텍스트를 `status.warnings`에 추가합니다.

임시 **PVC** 레이블 및 정리

Trident Protect는 그룹 스냅샷의 각 PVC에 레이블을 추가합니다(예: `protect.trident.netapp.io/vgs-group-673c0189-0=true`). 사용자가 지정한 레이블은 변경되지 않습니다. 레이블은 애플리케이션 스냅샷을 삭제할 때까지 PVC에 유지됩니다. 애플리케이션 스냅샷을 삭제하면 Trident Protect는 해당 볼륨 그룹 스냅샷과 멤버 스냅샷도 함께 삭제하고 PVC에서 레이블을 제거합니다. 이는 스냅샷의 `reclaimPolicy`이(가) `Delete`이든 `Retain`이든 상관없이 발생합니다.

다음 단계

다른 스냅샷과 마찬가지로 스냅샷에서 애플리케이션을 복원하십시오. 볼륨 그룹 스냅샷에는 별도의 복원 기능이 없습니다. Trident Protect는 멤버 스냅샷에서 복원하므로 그룹의 모든 PVC를 복원하지 않고도 개별 PVC 또는 VM을 복원할 수 있습니다. "스냅샷에서 다른 네임스페이스로 복구합니다" 또는 "스냅샷에서 원래 네임스페이스로 복구합니다"를 참조하십시오.

관련 정보

- "Trident Protect를 사용하여 관리를 위한 애플리케이션 정의"
- "Trident Protect를 사용하여 애플리케이션 보호"
- "KubeVirt 가상 머신을 Trident Protect로 보호합니다"

응용 프로그램을 복원합니다

Trident Protect를 사용하여 애플리케이션 복원

Trident Protect를 사용하면 스냅샷이나 백업에서 애플리케이션을 복원할 수 있습니다. 동일한 클러스터로 애플리케이션을 복원할 때 기존 스냅샷에서 복원하는 것이 더 빠릅니다.

참고

- Application Restore 시 해당 Application에 설정된 모든 execution hook이 App으로 복구된다. 복원 후 실행 후크가 있는 경우 복구 작업의 일부로 자동으로 실행됩니다.
- Qtree 볼륨의 경우 백업에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 기능이 지원됩니다. 그러나 Qtree 볼륨의 경우 스냅샷에서 다른 네임스페이스 또는 원래 네임스페이스로 복원하는 기능은 지원되지 않습니다.
- 고급 설정을 사용하여 복원 작업을 사용자 정의할 수 있습니다. 자세한 내용은 다음을 참조하세요. "고급 Trident Protect 복원 설정 사용".
- 모든 네임스페이스 리소스를 복원하지 않고 아카이브에서 특정 KubeVirt 가상 머신(VM)을 복원하려면 복원 CR의 `includedVirtualMachines` 필드를 사용하십시오. 지원되는 CR, 필드 참조 및 YAML 예제는 "KubeVirt 가상 머신을 Trident Protect로 보호합니다"을 참조하십시오.

백업에서 다른 네임스페이스로 복원합니다

BackupRestore CR을 사용하여 다른 네임스페이스로 백업을 복원하면 Trident Protect는 새 네임스페이스에서 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 만듭니다. 복구된 애플리케이션을 보호하려면 주문형 백업이나 스냅샷을 만들거나 보호 일정을 설정하세요.

참고

- 기존 리소스가 있는 다른 네임스페이스로 백업을 복원해도 백업의 리소스와 이름을 공유하는 리소스는 변경되지 않습니다. 백업의 모든 리소스를 복구하려면 타겟 네임스페이스를 삭제하고 다시 생성하거나 백업을 새 네임스페이스로 복원하십시오.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 대상 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS IAM 설명서" 참조하십시오.

참고

Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-restore-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appArchivePath * : 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - * spec.appVaultRef *: (*required*) 백업 내용이 저장된 AppVault의 이름입니다.
   - **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
   - **spec.namespaceMapping**: 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace`및 를 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

     ```
     ---
     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
       destinationApplicationName: my-new-app-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

   참고 Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

   - **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) `Include` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 `Exclude` 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.

- **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
  - **resourceMatchers[].group**:(*Optional*) 필터링할 리소스의 그룹입니다.
  - * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
  - **resourceMatchers [ ].version**: (*Optional*) 필터링할 리소스의 버전입니다.
  - **resourceMatchers[].names**:(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
  - *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - **resourceMatchers [ ].labelSelectors**: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "Kubernetes 문서" 예를 들면 다음과 `"trident.netapp.io/os=linux"`같습니다.

    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. 파일을 올바른 값으로 채운 후 `trident-protect-backup-restore-cr.yaml` CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 환경의 정보로 대체하여 백업을 다른 네임스페이스로 복원합니다. 이 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

백업에서 원래 네임스페이스로 복구합니다

언제든지 백업을 원래 네임스페이스로 복원할 수 있습니다. 제자리 복원을 수행하면 Trident Protect는 잘못된 복구 지점이 생성되는 것을 방지하기 위해 보호 일정과 진행 중인 작업을 자동으로 관리합니다.

- 복원 작업이 시작되기 전에 애플리케이션에 대해 활성화된 모든 보호 일정이 비활성화됩니다. 이렇게 하면 애플리케이션 리소스가 복원되는 동안 예약된 백업이나 스냅샷이 실행되는 것을 방지할 수 있습니다.
- 복원 작업이 성공적으로 완료되면 복원 전에 활성화되어 있던 일정만 다시 활성화됩니다. 이미 비활성화된 일정은 계속 비활성화된 상태로 유지됩니다.
- 복원이 시작되기 전에 진행 중인 모든 백업 또는 스냅샷 작업이 취소됩니다. 5분 이내에 작업이 취소되지 않으면 복원이 진행되고 복원 CR 상태에 경고가 기록됩니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS IAM 설명서" 참조하십시오.

참고	Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-ipr-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appArchivePath * : 백업 컨텐츠가 저장되는 AppVault 내부의 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

◦ * spec.appVaultRef *: (*required*) 백업 내용이 저장된 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) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

참고	Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

◦ **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) `Include` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 `Exclude` 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.

- **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
  - **resourceMatchers[].group**:(*Optional*) 필터링할 리소스의 그룹입니다.
  - * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
  - **resourceMatchers [ ].version**: (*Optional*) 필터링할 리소스의 버전입니다.
  - **resourceMatchers[].names**:(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
  - *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
  - **resourceMatchers [ ].labelSelectors**: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "Kubernetes 문서" 예를 들면 다음과 `"trident.netapp.io/os=linux"`같습니다.

    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. 파일을 올바른 값으로 채운 후 `trident-protect-backup-ipr-cr.yaml` CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 환경의 정보로 대체하여 백업을 원래 네임스페이스로 복원합니다. `backup`인수에 네임스페이스 및 백업 이름이 형식으로 `<namespace>/<name>` 사용됩니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

백업에서 다른 클러스터로 복원합니다

원래 클러스터에 문제가 있는 경우 백업을 다른 클러스터로 복구할 수 있습니다.

참고

- Kopia를 데이터 이동자로 사용하여 백업을 복원하는 경우 CR이나 CLI에서 주석을 지정하여 Kopia에서 사용하는 임시 저장소의 동작을 제어할 수 있습니다. 를 참조하세요 "코피아 문서" 구성할 수 있는 옵션에 대한 자세한 내용은 다음을 참조하세요. 사용하다 `tridentctl-protect create --help` Trident Protect CLI로 주석을 지정하는 방법에 대한 자세한 내용은 명령을 참조하세요.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 대상 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

다음 필수 구성 요소가 충족되는지 확인합니다.

- 대상 클러스터에 Trident Protect가 설치되어 있습니다.
- 대상 클러스터에는 백업이 저장되는 소스 클러스터와 동일한 AppVault의 버킷 경로에 대한 액세스 권한이 있습니다.
- AppVault CR에서 정의된 개체 저장소 버킷에 로컬 환경이 연결할 수 있는지 확인하십시오. `tridentctl-protect get appvaultcontent` 명령. 네트워크 제한으로 인해 액세스가 불가능한 경우 대상 클러스터의 포드 내에서 Trident Protect CLI를 실행하세요.
- AWS 세션 토큰 만료가 장기 실행 중인 복구 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.
  - 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
  - AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS 설명서" 참조하십시오.

단계

1. Trident Protect CLI 플러그인을 사용하여 타겟 클러스터에 AppVault CR이 있는지 확인합니다.

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

참고	타겟 클러스터에 AppVault CR이 없는 경우 "Trident Protect AppVault 객체를 사용하여 버킷을 관리합니다."의 단계에 따라 생성하십시오.

2. 타겟 클러스터에서 사용 가능한 AppVault의 백업 콘텐츠를 확인하고 복원할 백업의 `appArchivePath`을 기록해 두십시오.

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

이 명령을 실행하면 원래 클러스터, 해당 응용 프로그램 이름, 타임스탬프 및 아카이브 경로를 비롯하여 AppVault에 사용 가능한 백업이 표시됩니다.

  - 출력 예: *

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   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 이름 및 보관 경로를 사용하여 응용 프로그램을 대상 클러스터로 복원합니다.

참고	CR을 사용할 때는 애플리케이션 복원에 사용할 네임스페이스가 타겟 클러스터에 있는지 확인하십시오.

CR을 사용합니다

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-backup-restore-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appVaultRef *: (*required*) 백업 내용이 저장된 AppVault의 이름입니다.
   - **spec.appArchivePath**: (필수) AppVault 내에서 백업 콘텐츠가 저장되는 경로입니다. 2단계의 명령을 사용하여 백업 콘텐츠를 확인하고 `appArchivePath`복원하려는 백업을 찾으십시오.
   - **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
   - **spec.namespaceMapping**: 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace`및 를 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

     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
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 파일을 올바른 값으로 채운 후 `trident-protect-backup-restore-cr.yaml` CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

CLI를 사용합니다

1. 다음 명령을 사용하여 응용 프로그램을 복원하고 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

네임스페이스 매핑 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 Source1:dest1, source2:dest2 형식으로 올바른 대상 네임스페이스에 매핑합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

스냅샷에서 다른 네임스페이스로 복구합니다

사용자 정의 리소스(CR) 파일을 사용하여 스냅샷에서 데이터를 다른 네임스페이스나 원래 소스 네임스페이스로 복원할 수 있습니다. SnapshotRestore CR을 사용하여 다른 네임스페이스로 스냅샷을 복원하면 Trident Protect는 새 네임스페이스에서 애플리케이션을 복원하고 복원된 애플리케이션에 대한 애플리케이션 CR을 만듭니다. 복구된 애플리케이션을 보호하려면 주문형 백업이나 스냅샷을 만들거나 보호 일정을 설정하세요.

참고

- SnapshotRestore는 다음을 지원합니다. `spec.storageClassMapping` 속성이지만 소스 및 대상 저장소 클래스가 동일한 저장소 백엔드를 사용하는 경우에만 해당됩니다. 복원을 시도하는 경우 `StorageClass` 다른 스토리지 백엔드를 사용하는 경우 복원 작업이 실패합니다.
- CR을 사용하여 새 네임스페이스로 복원할 경우, CR을 적용하기 전에 대상 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS IAM 설명서" 참조하십시오.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-restore-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appVaultRef *: (*required*) 스냅샷 내용이 저장된 AppVault의 이름입니다.
   - * spec.appArchivePath *: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     ```

```
jsonpath='{.status.appArchivePath}'
```

- **spec.destinationApplicationName**: (선택 사항) 복원된 애플리케이션의 이름입니다. 제공된 경우 복원된 애플리케이션은 이 이름을 사용합니다. 제공되지 않은 경우 복원된 애플리케이션은 원본 애플리케이션 이름을 사용합니다.
- **spec.namespaceMapping**: 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace`및 를 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. (선택 사항) 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.

참고	Trident Protect는 사용자가 선택한 리소스와의 관계로 인해 일부 리소스를 자동으로 선택합니다. 예를 들어, 영구 볼륨 클레임 리소스를 선택하고 연관된 Pod가 있는 경우 Trident Protect는 연관된 Pod도 복원합니다.

- **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) `Include` resourceMatchers에 정의된 리소스를 포함하거나 제외하려면 또는 을 `Exclude` 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
  - **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
    - **resourceMatchers[].group**:(*Optional*) 필터링할 리소스의 그룹입니다.
    - * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
    - **resourceMatchers [ ].version**: (*Optional*) 필터링할 리소스의 버전입니다.
    - **resourceMatchers[].names**:(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
    - *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
    - **resourceMatchers [ ].labelSelectors**: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "Kubernetes 문서" 예를 들면 다음과 `"trident.netapp.io/os=linux"`같습니다.

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. 파일을 올바른 값으로 채운 후 `trident-protect-snapshot-restore-cr.yaml` CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 다른 네임스페이스로 복원합니다.

   - `snapshot`인수에 네임스페이스 및 스냅숏 이름이 형식으로 `<namespace>/<name>` 사용됩니다.
   - 이 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다.

   예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

스냅샷에서 원래 네임스페이스로 복구합니다

언제든지 스냅샷을 원래 네임스페이스로 복원할 수 있습니다. 제자리 복원을 수행하면 Trident Protect는 잘못된 복구 지점이 생성되는 것을 방지하기 위해 보호 일정과 진행 중인 작업을 자동으로 관리합니다.

- 복원 작업이 시작되기 전에 애플리케이션에 대해 활성화된 모든 보호 일정이 비활성화됩니다. 이렇게 하면 애플리케이션 리소스가 복원되는 동안 예약된 백업이나 스냅샷이 실행되는 것을 방지할 수 있습니다.
- 복원 작업이 성공적으로 완료되면 복원 전에 활성화되어 있던 일정만 다시 활성화됩니다. 이미 비활성화된 일정은 계속 비활성화된 상태로 유지됩니다.
- 복원이 시작되기 전에 진행 중인 모든 백업 또는 스냅샷 작업이 취소됩니다. 5분 이내에 작업이 취소되지 않으면 복원이 진행되고 복원 CR 상태에 경고가 기록됩니다.

시작하기 전에

AWS 세션 토큰의 만료가 장시간 실행되는 S3 복원 작업에 충분한지 확인합니다. 복구 작업 중에 토큰이 만료되면 작업이 실패할 수 있습니다.

- 현재 세션 토큰 만료 확인에 대한 자세한 내용은 를 "AWS API 설명서" 참조하십시오.
- AWS 리소스의 자격 증명에 대한 자세한 내용은 를 "AWS IAM 설명서" 참조하십시오.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-ipr-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appVaultRef *: (*required*) 스냅샷 내용이 저장된 AppVault의 이름입니다.
   - * spec.appArchivePath *: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
```

+ . (선택 사항) 애플리케이션의 특정 리소스만 복원해야 하는 경우, 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가하세요.

+ 참고: 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 `resourceMatchers`에 정의된 리소스를 포함하거나 제외합니다. 포함 또는 제외할 리소스를 정의하려면 다음 `resourceMatchers` 매개변수를 추가합니다.
resourceFilter.resourceMatchers: **resourceMatcher** 객체의 배열입니다. 이 배열에 여러 요소를 정의하는 경우 **OR** 연산으로 일치하고, 각 요소 내부의 필드**(group, kind, version)**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:

+

```
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"]
```

+ `trident-protect-snapshot-ipr-cr.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

+

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 원래 네임스페이스로 복원합니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

복구 작업의 상태를 확인합니다

명령줄을 사용하여 진행 중이거나, 완료되었거나, 실패한 복구 작업의 상태를 확인할 수 있습니다.

단계

1. 다음 명령을 사용하여 복원 작업의 상태를 검색하여 대괄호의 값을 사용자 환경의 정보로 바꿉니다.

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

고급 **Trident Protect** 복원 설정 사용

주석, 네임스페이스 설정, 스토리지 옵션 등의 고급 설정을 사용하여 복원 작업을 사용자 정의하여 특정 요구 사항을 충족할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석이 소스 네임스페이스의 레이블 및 주석과 일치하도록 만듭니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블 또는 주석이 추가되고 이미 존재하는 모든 레이블 또는 주석이 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 있는 레이블이나 주석은 변경되지 않습니다.

참고

Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요."OpenShift 보안 컨텍스트 제약 조건 문서" .

복원 또는 페일오버 작업을 수행하기 전에 Kubernetes 환경 변수를 설정하여 타겟 네임스페이스의 특정 주석을 덮어쓰지 않도록 할 수 있습니다 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`. 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
```

참고

복원 또는 장애 조치 작업을 수행할 때 지정된 모든 네임스페이스 주석 및 레이블 `restoreSkipNamespaceAnnotations` 그리고 `restoreSkipNamespaceLabels` 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한

내용은 다음을 참조하세요. "추가 Trident Protect 헬름 차트 설정 구성".

Helm을 사용하여 소스 애플리케이션을 설치한 경우 `--create-namespace` 깃발에는 특별한 대우가 주어집니다. `name` 라벨 키. 복구 또는 장애 조치 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스 네임스페이스와 일치하지 않으면 변경 사항 없이 대상 네임스페이스에 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여 줍니다. 작업 전후에 대상 네임스페이스의 상태와 대상 네임스페이스에서 주석과 레이블이 결합되거나 덮어써지는 방법을 확인할 수 있습니다.

복구 또는 페일오버 작업 전

다음 표에서는 복구 또는 페일오버 작업 이전의 예제 소스 및 대상 네임스페이스의 상태를 보여 줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키):	• 환경 = 운영 • 규정 준수 = HIPAA • 이름 = ns-1
네임스페이스 ns-2(대상)	• Annotation.one/key(주석. 하나/키):"true" • Annotation.Three/key:"false"	• 역할 = 데이터베이스

복구 작업 후

다음 표에서는 복구 또는 페일오버 작업 후의 예제 대상 네임스페이스의 상태를 보여 줍니다. 일부 키가 추가되고, 일부 키가 덮어쓰여졌으며, `name` 대상 네임스페이스와 일치하도록 레이블이 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키): • Annotation.Three/key:"false"	• 이름 = ns-2 • 규정 준수 = HIPAA • 환경 = 운영 • 역할 = 데이터베이스

지원되는 필드

이 섹션에서는 복원 작업에 사용할 수 있는 추가 필드에 대해 설명합니다.

스토리지 클래스 매핑

그만큼 `spec.storageClassMapping` 속성은 소스 애플리케이션에 있는 스토리지 클래스에서 대상 클러스터의 새 스토리지 클래스로의 매핑을 정의합니다. 서로 다른 스토리지 클래스를 사용하는 클러스터 간에 애플리케이션을 마이그레이션하거나 BackupRestore 작업에 대한 스토리지 백엔드를 변경할 때 이 기능을 사용할 수 있습니다.

- 예: *

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

지원되는 주석

이 섹션에서는 시스템의 다양한 동작을 구성하는 데 지원되는 애노테이션을 나열합니다. 사용자가 애노테이션을 명시적으로 설정하지 않으면 시스템은 기본값을 사용합니다.

주석	유형	설명	기본값
보호.트라이던트.넷앱.io/데이터-무버-타임아웃-초	문자열	데이터 이동 작업이 중단되는 데 허용되는 최대 시간(초)입니다.	"300"
보호.트라이던트.넷앱.io/코피아-콘텐츠-캐시-크기-제한-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 도달할 때까지 기다리는 최대 시간(초) `Bound` 작업이 실패하기 전 단계입니다. 모든 복원 CR 유형(BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore)에 적용됩니다. 스토리지 백엔드나 클러스터에 더 많은 시간이 필요한 경우 더 높은 값을 사용하세요.	"1200"(20분)
protect.trident.netapp.io/retain-mac-address	문자열	KubeVirt VM의 경우, 복원 시 첫 번째 네트워크 인터페이스의 MAC 주소를 유지하려면(예: 애플리케이션 라이선스가 해당 인터페이스에 바인딩된 경우) `"true"`로 설정하십시오. 모든 복원 CR 유형 및 AppMirrorRelationship에 적용됩니다(관계를 승격하기 전에 설정하십시오). 기본적으로 Trident Protect는 충돌을 방지하기 위해 이 MAC 주소를 제거합니다. 소스 VM이 동일한 네트워크에서 계속 실행 중인 경우, 중복 MAC 주소를 방지하기 위해 먼저 종료하거나 격리된 네트워크로 복원하십시오.	설정되지 않음(MAC 주소가 삭제됨)

사용자 지정 리소스를 사용하여 리소스를 복원할 때 리소스를 수정합니다

리소스 변환을 사용하면 리소스를 복원하는 동안 수정할 수 있습니다. 이는 복원된 버전이 원본과 달라야 하는 경우, 예를 들어 애플리케이션을 다른 클러스터로 복원할 때 ConfigMap의 호스트 이름을 업데이트하는 경우에 유용합니다.

리소스 수정 작동 방식

`SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` 및 기타 복구 리소스에 있는 `transformations` 필드를 사용하면 복구 프로세스 중에 Kubernetes

리소스를 수정할 수 있습니다. 이는 예를 들어 호스트 이름, 레지스트리 URL, 리소스 제한 또는 환경 변수 변경과 같이 복구된 애플리케이션을 원본과 다르게 구성해야 할 때 유용합니다.

리소스 변환은 "RFC 6902 JSON Patch" 작업과 "RFC 6901 JSON 포인터" 경로를 사용하여 Kubernetes 리소스 내의 특정 필드를 대상으로 지정하고 수정합니다.

다음은 리소스 변환을 포함하는 복원 객체의 기본 구조를 보여줍니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
resources)
        version: ""           # Optional: API version
        name: ""              # Optional: specific resource name
      operations:
        - op: replace         # Operation type
          path: "/data/key"   # JSON Pointer path
          value: "new-value"  # New value (for add/replace/test)
```

지원되는 리소스

다음 기준과 일치하는 리소스에 대해 리소스 변환을 사용할 수 있습니다.

- **kind** (필수): Kubernetes 리소스 종류 (예: `ConfigMap`, `Deployment`, `Pod`)
- 그룹 (선택 사항): API 그룹(예: `apps`, `route.openshift.io`) - 핵심 리소스의 경우 생략
- 버전 (선택 사항): API 버전(예: `v1`, `v1beta1`)
- **name** (선택 사항): 이름으로 지정된 특정 리소스에만 적용합니다.

참고 PersistentVolumeClaims 및 네임스페이스 수정은 현재 지원되지 않습니다.

지원되는 작업

리소스를 수정하려면 다음 작업을 사용할 수 있습니다. `add`, `copy`, `move`, `remove`, `replace`, 및 `test`.

리소스에 값 추가

`add` 작업을 사용하여 지정된 경로에 새 필드 또는 값을 추가합니다. 객체 또는 배열에 데이터를 추가할 수 있습니다.

CR을 사용합니다

다음 예제는 a `Deployment` 리소스에 노드 선택기를 추가합니다.

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

리소스 내의 값 복사

`copy` 연산을 사용하여 동일한 리소스 내의 한 경로에서 다른 경로로 값을 복사할 수 있습니다. 원본은 변경되지 않습니다.

CR을 사용합니다

다음 예제는 `ConfigMap` 객체의 데이터 키를 복제합니다.

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

리소스 내에서 값 이동

`move` 연산을 사용하여 동일한 리소스 내에서 한 경로에서 다른 경로로 값을 이동할 수 있습니다. 원본은 제거되고 값은 대상에 배치됩니다.

CR을 사용합니다

다음 예제는 `ConfigMap` 객체의 데이터 키 이름을 변경합니다.

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
```

```
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

리소스에서 값 제거

`remove` 작업을 사용하여 지정된 경로에서 필드 또는 값을 제거합니다.

CR을 사용합니다

다음 예제는 ConfigMap 리소스에서 주석을 제거합니다.

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

참고 위 예시의 경로에서 `~1`는 `/`에 대한 JSON 포인터 이스케이프 시퀀스입니다.

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
io~1last-applied-configuration"}'
```

리소스 내의 값을 바꿉니다

`replace` 작업을 사용하여 지정된 경로에 있는 리소스 내의 기존 값을 바꿉니다. JSON 경로는 이미 존재해야 합니다.

CR을 사용합니다

다음 예제는 Route 객체의 호스트 이름을 변경합니다.

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"
```

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

리소스 수정 테스트

이 `test` 연산을 사용하여 다른 연산을 적용하기 전에 경로의 값이 예상 값과 일치하는지 확인하십시오. 테스트에 실패하면 전체 수정 사항이 롤백됩니다.

CR을 사용합니다

다음 예는 `environment`이 `staging`(으)로 설정된 경우에만 `database-host`을 업데이트합니다:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

CLI를 사용합니다

명령줄을 사용하여 이 변환을 실행하려면 다음 명령을 사용하십시오.

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

NetApp SnapMirror를 사용하여 애플리케이션 복제

Trident Protect를 사용하여 애플리케이션 복제를 설정하고 관리합니다.

Trident Protect를 사용하면 NetApp SnapMirror 기술의 비동기 복제 기능을 사용하여 동일한 클러스터 또는 서로 다른 클러스터 간에 데이터 및 애플리케이션 변경 사항을 한 스토리지 백엔드에서 다른 스토리지 백엔드로 복제할 수 있습니다.

복원 및 페일오버 작업 중 네임스페이스 주석 및 레이블

복원 및 페일오버 작업 중에 대상 네임스페이스의 레이블과 주석이 소스 네임스페이스의 레이블 및 주석과 일치하도록 만듭니다. 대상 네임스페이스에 없는 소스 네임스페이스의 레이블 또는 주석이 추가되고 이미 존재하는 모든 레이블 또는 주석이 소스 네임스페이스의 값과 일치하도록 덮어쓰여집니다. 대상 네임스페이스에만 있는 레이블이나 주석은 변경되지 않습니다.

참고	Red Hat OpenShift를 사용하는 경우 OpenShift 환경에서 네임스페이스 주석의 중요한 역할을 알아두는 것이 중요합니다. 네임스페이스 주석은 복원된 포드가 OpenShift 보안 컨텍스트 제약(SCC)에 의해 정의된 적절한 권한과 보안 구성을 준수하고 권한 문제 없이 볼륨에 액세스할 수 있도록 보장합니다. 자세한 내용은 다음을 참조하세요."OpenShift 보안 컨텍스트 제약 조건 문서" .

복원 또는 페일오버 작업을 수행하기 전에 Kubernetes 환경 변수를 설정하여 타겟 네임스페이스의 특정 주석을 덮어쓰지 않도록 할 수 있습니다 `RESTORE_SKIP_NAMESPACE_ANNOTATIONS`. 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
```

참고	복원 또는 장애 조치 작업을 수행할 때 지정된 모든 네임스페이스 주석 및 레이블 `restoreSkipNamespaceAnnotations` 그리고 `restoreSkipNamespaceLabels` 복구 또는 장애 조치 작업에서 제외됩니다. 이러한 설정은 Helm을 처음 설치할 때 구성되어야 합니다. 자세한 내용은 다음을 참조하세요. "추가 Trident Protect 헬름 차트 설정 구성".

Helm을 사용하여 소스 애플리케이션을 설치한 경우 `--create-namespace` 깃발에는 특별한 대우가 주어집니다. `name` 라벨 키. 복구 또는 장애 조치 프로세스 중에 Trident Protect는 이 레이블을 대상 네임스페이스에 복사하지만 소스의 값이 소스 네임스페이스와 일치하는 경우 값을 대상 네임스페이스 값으로 업데이트합니다. 이 값이 소스

네임스페이스와 일치하지 않으면 변경 사항 없이 대상 네임스페이스에 복사됩니다.

예

다음 예제에서는 각각 다른 주석과 레이블이 있는 소스 및 대상 네임스페이스를 보여 줍니다. 작업 전후에 대상 네임스페이스의 상태와 대상 네임스페이스에서 주석과 레이블이 결합되거나 덮어써지는 방법을 확인할 수 있습니다.

복구 또는 페일오버 작업 전

다음 표에서는 복구 또는 페일오버 작업 이전의 예제 소스 및 대상 네임스페이스의 상태를 보여 줍니다.

네임스페이스	주석	라벨
네임스페이스 ns-1(소스)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키):	• 환경 = 운영 • 규정 준수 = HIPAA • 이름 = ns-1
네임스페이스 ns-2(대상)	• Annotation.one/key(주석. 하나/키):"true" • Annotation.Three/key:"false"	• 역할 = 데이터베이스

복구 작업 후

다음 표에서는 복구 또는 페일오버 작업 후의 예제 대상 네임스페이스의 상태를 보여 줍니다. 일부 키가 추가되고, 일부 키가 덮어쓰여졌으며, `name` 대상 네임스페이스와 일치하도록 레이블이 업데이트되었습니다.

네임스페이스	주석	라벨
네임스페이스 ns-2(대상)	• annotation.one/key:"updatedvalue"(주석 1개/키) • Annotation.two/key(주석.two/키): • Annotation.Three/key:"false"	• 이름 = ns-2 • 규정 준수 = HIPAA • 환경 = 운영 • 역할 = 데이터베이스

참고 데이터 보호 작업 중에 파일 시스템을 동결 및 동결 해제하도록 Trident Protect를 구성할 수 있습니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".

장애 조치 및 역방향 작업 중 실행 후크

AppMirror 관계를 사용하여 애플리케이션을 보호하는 경우 장애 조치 및 역방향 작업 중에 알고 있어야 할 실행 후크와 관련된 특정 동작이 있습니다.

- 장애 조치(failover) 중에 실행 후크는 소스 클러스터에서 대상 클러스터로 자동으로 복사됩니다. 따라서 수동으로 다시 만들 필요가 없습니다. 장애 조치(failover) 후에는 실행 후크가 애플리케이션에 존재하며 관련 작업 중에 실행됩니다.
- 역방향 또는 역방향 재동기화 중에는 애플리케이션에 있는 기존 실행 후크가 모두 제거됩니다. 소스 애플리케이션이 대상 애플리케이션으로 변경되면 이러한 실행 후크는 유효하지 않으므로 삭제되어 실행되지 않습니다.

실행 후크에 대해 자세히 알아보려면 다음을 참조하세요."Trident Protect 실행 후크 관리" .

복제 관계를 설정하려면 다음을 수행해야 합니다.

- Trident Protect가 앱 스냅샷을 얼마나 자주 찍을지 선택합니다(여기에는 앱의 Kubernetes 리소스와 각 앱 볼륨에 대한 볼륨 스냅샷이 포함됨)
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 생성할 시간을 설정합니다

단계

1. 소스 클러스터에서 소스 응용 프로그램에 대한 AppVault를 생성합니다. 스토리지 공급자에 따라 의 예를 "AppVault 사용자 지정 리소스"환경에 맞게 수정합니다.

CR을 사용하여 **AppVault**를 작성합니다

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-appvault-primary-source.yaml`).

b. 다음 특성을 구성합니다.

- **metadata.name**: (*required*) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
- **spec.providerConfig**: (*required*) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. bucketName과 공급자에게 필요한 기타 세부 정보를 선택합니다. 복제 관계에 필요한 다른 CR 파일은 이러한 값을 참조하므로 선택한 값을 기록해 둡니다. 다른 공급자의 AppVault CRS에 대한 예는 을 "AppVault 사용자 지정 리소스"참조하십시오.
- **spec.providerCredentials**: (*required*) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
  - **spec.providerCredentials.valueFromSecret**: (*required*) 자격 증명 값이 비밀에서 와야 함을 나타냅니다.
    - * key *: (*required*) 선택할 비밀의 유효한 키입니다.
    - * name *: (*required*) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.
  - **spec.providerCredentials.secretAccessKey**: (*required*) 제공자에 액세스하는 데 사용되는 액세스 키입니다. name * 은 * spec.providerCredentials.valueFromSecret.name* 과 일치해야 합니다.
- **spec.providerType**: (*required*) 백업을 제공하는 항목을 결정합니다(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure). 가능한 값:
  - 설치하고
  - Azure를 지원합니다
  - GCP
  - 일반 - S3
  - ONTAP-S3
  - StorageGRID-S3

c. 파일을 올바른 값으로 채운 후 `trident-protect-appvault-primary-source.yaml` CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

CLI를 사용하여 **AppVault**를 작성합니다

a. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 AppVault를 작성합니다.

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. 소스 클러스터에서 소스 애플리케이션 CR:

CR을 사용하여 소스 응용 프로그램을 만듭니다

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-app-source.yaml`).

b. 다음 특성을 구성합니다.

- **metadata.name**: (*required*) 응용 프로그램 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
- **spec.includedNamespaces**: (*required*) 네임스페이스 및 관련 레이블의 배열입니다. 네임스페이스 이름을 사용하고 선택적으로 레이블을 사용하여 네임스페이스 범위를 좁혀 여기에 나열된 네임스페이스에 있는 리소스를 지정합니다. 응용 프로그램 네임스페이스는 이 배열의 일부여야 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. 파일을 올바른 값으로 채운 후 `trident-protect-app-source.yaml` CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-
namespace
```

CLI를 사용하여 소스 애플리케이션을 생성합니다

a. 소스 응용 프로그램을 만듭니다. 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 만듭니다. 이 스냅샷은 대상 클러스터의 애플리케이션의 기반으로 사용됩니다. 이 단계를 건너뛰면 다음 예약된 스냅샷이 실행될 때까지 기다려야 최신 스냅샷을 얻을 수 있습니다. 주문형 스냅샷을 생성하려면 다음을 참조하세요. "필요 시 스냅샷을 생성합니다".
4. 소스 클러스터에서 복제 일정 CR을 만듭니다.

참고

아래 제공된 일정 외에도, 피어링된 ONTAP 클러스터 간에 공통 스냅샷을 유지하기 위해 7일의 보존 기간을 갖는 별도의 일일 스냅샷 일정을 생성하는 것이 좋습니다. 이를 통해 스냅샷을 최대 7일 동안 사용할 수 있지만, 보존 기간은 사용자 요구 사항에 따라 맞춤 설정할 수 있습니다.

장애 조치(failover)가 발생하면 시스템은 최대 7일 동안 이러한 스냅샷을 역방향 작업에 사용할 수 있습니다. 이 방식은 마지막 스냅샷 이후 변경된 내용만 전송하고 모든 데이터는 전송하지 않으므로 역방향 프로세스가 더 빠르고 효율적입니다.

해당 애플리케이션의 기존 일정이 이미 원하는 보존 요구 사항을 충족하는 경우 추가 일정은 필요하지 않습니다.

CR을 사용하여 복제 일정을 만듭니다**.**

a. 소스 애플리케이션에 대한 복제 스케줄을 생성합니다.

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-schedule.yaml`).

ii. 다음 특성을 구성합니다.

- **metadata.name**: (*required*) 일정 사용자 정의 리소스의 이름입니다.
- **spec.appVaultRef**: (필수) 이 값은 소스 애플리케이션의 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.applicationRef**: (필수) 이 값은 소스 애플리케이션 CR의 metadata.name 필드와 일치해야 합니다.
- **spec.backupRetention**: (*required*) 이 필드는 필수 필드이며 값을 0으로 설정해야 합니다.
- * spec. enabled *: 반드시 true로 설정해야 합니다.
- **spec.granularity**: 으로 설정해야 `Custom` 합니다.
- **spec.recurrenceRule**: UTC 시간과 반복 간격으로 시작 날짜를 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. 파일을 올바른 값으로 채운 후 `trident-protect-schedule.yaml` CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

CLI를 사용하여 복제 일정을 만듭니다.

a. 괄호 안의 값을 사용자 환경의 정보로 바꿔 복제 일정을 만듭니다.

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

- 예: *

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. 대상 클러스터에서 소스 클러스터에 적용한 AppVault CR과 동일한 소스 응용 프로그램 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. 대상 클러스터에서 대상 응용 프로그램에 대한 대상 AppVault CR을 생성합니다. 스토리지 공급자에 따라 의 예를 "AppVault 사용자 지정 리소스"환경에 맞게 수정합니다.

   a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-appvault-secondary-destination.yaml`).

   b. 다음 특성을 구성합니다.

      - **metadata.name**: (*required*) AppVault 사용자 정의 리소스의 이름입니다. 복제 관계에 필요한 다른 CR 파일은 이 값을 참조하므로 선택한 이름을 기록해 둡니다.
      - **spec.providerConfig**: (*required*) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 구성을 저장합니다. `bucketName`와 공급자에게 필요한 기타 세부 정보를 선택합니다. 복제 관계에 필요한 다른 CR 파일은 이러한 값을 참조하므로 선택한 값을 기록해 둡니다. 다른 공급자의 AppVault CRS에 대한 예는 을 "AppVault 사용자 지정 리소스"참조하십시오.
      - **spec.providerCredentials**: (*required*) 지정된 공급자를 사용하여 AppVault에 액세스하는 데 필요한 자격 증명에 대한 참조를 저장합니다.
        - **spec.providerCredentials.valueFromSecret**: (*required*) 자격 증명 값이 비밀에서 와야 함을 나타냅니다.
          - * key *: (*required*) 선택할 비밀의 유효한 키입니다.
          - * name *: (*required*) 이 필드의 값을 포함하는 비밀의 이름입니다. 같은 네임스페이스에 있어야 합니다.
        - **spec.providerCredentials.secretAccessKey**: (*required*) 제공자에 액세스하는 데 사용되는 액세스 키입니다. name * 은 * spec.providerCredentials.valueFromSecret.name* 과 일치해야 합니다.
      - **spec.providerType**: (*required*) 백업을 제공하는 항목을 결정합니다(예: NetApp ONTAP S3, 일반 S3, Google Cloud 또는 Microsoft Azure). 가능한 값:
        - 설치하고
        - Azure를 지원합니다
        - GCP
        - 일반 - S3
        - ONTAP-S3
        - StorageGRID-S3

   c. 파일을 올바른 값으로 채운 후 `trident-protect-appvault-secondary-destination.yaml` CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. 대상 클러스터에서 AppMirrorRelationship CR 파일을 생성합니다.

참고	CR을 사용할 때는 CR을 적용하기 전에 대상 네임스페이스를 수동으로 생성해야 합니다. Trident Protect는 CLI를 사용할 때만 네임스페이스를 자동으로 생성합니다.

CR을 사용하여 **AppMirrorRelationship**을 생성합니다

a. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-relationship.yaml`).

b. 다음 특성을 구성합니다.

- **metadata.name:** (필수) AppMirrorRelationship 사용자 정의 리소스의 이름입니다.
- **metadata.annotations.protect.trident.netapp.io/retain-mac-address**: (선택 사항) 타겟 클러스터에서 KubeVirt VM의 MAC 주소를 유지하려면 `"true"`(으)로 설정합니다. "고급 Trident Protect 복원 설정 사용"을(를) 참조하십시오.
- **spec.destinationAppVaultRef**: (*required*) 이 값은 대상 클러스터의 대상 응용 프로그램에 대한 AppVault 이름과 일치해야 합니다.
- **spec.namespaceMapping**: (*required*) 대상 및 소스 네임스페이스는 해당 응용 프로그램 CR에 정의된 응용 프로그램 네임스페이스와 일치해야 합니다.
- * spec.sourceAppVaultRef *: (*required*) 이 값은 소스 응용 프로그램의 AppVault 이름과 일치해야 합니다.
- * spec.sourceApplicationName *: (*required*) 이 값은 소스 응용 프로그램 CR에서 정의한 소스 응용 프로그램의 이름과 일치해야 합니다.
- **spec.sourceApplicationUID**: (필수) 이 값은 소스 애플리케이션 CR에서 정의한 소스 애플리케이션의 UID와 일치해야 합니다.
- **spec.storageClassName**: (선택 사항) 클러스터에서 유효한 스토리지 클래스의 이름을 선택합니다. 스토리지 클래스는 소스 환경과 피어링된 ONTAP 스토리지 VM에 연결되어야 합니다. 스토리지 클래스가 제공되지 않으면 기본적으로 클러스터의 기본 스토리지 클래스가 사용됩니다.
- **spec.storageClassMapping**: (선택 사항) 동일한 AMR 내의 개별 PVC에 대해 소스 스토리지 클래스를 타겟 스토리지 클래스에 매핑합니다. 각 항목에는 `source` 및 `destination` 스토리지 클래스 이름이 필요합니다. 설정된 경우, `storageClassMapping`은(는) `storageClassName`을(를) 완전히 재정의하며, 일치하는 소스 항목이 없는 PVC는 `storageClassName`이(가) 아닌 클러스터 기본 스토리지 클래스를 사용합니다.

  `storageClassMapping`을 AppMirrorRelationship CR에서 사용할 때는 다음과 같은 제한 사항에 유의하십시오.

  - `storageClassMapping`이(가) `storageClassName`을(를) 완전히 덮어씁니다. 두 필드가 모두 설정된 경우 `storageClassName`은(는) 무시됩니다.
  - 에 일치하는 항목이 없는 소스 스토리지 클래스를 가진 PVC는 `storageClassMapping` 클러스터 기본 스토리지 클래스를 사용하며, `storageClassName`은(는) 사용하지 않습니다.
  - 각 항목에는 `source`와 `destination` 값이 모두 필요합니다.
  - 대상 스토리지 클래스를 변경하면 기존 대상 PVC가 삭제되고 다시 생성될 수 있습니다.
- **spec.recurrenceRule**: UTC 시간과 반복 간격으로 시작 날짜를 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. 파일을 올바른 값으로 채운 후 `trident-protect-relationship.yaml` CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

CLI를 사용하여 **AppMirrorRelationship**을 생성한다

a. AppMirrorRelationship 객체를 만들고 적용하며, 괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirrorrelationship> --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> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

참고

`--storageclass-mapping`을 사용하여 개별 소스 스토리지 클래스를 대상 스토리지 클래스에 매핑합니다. 쉼표로 구분하여 하나 이상의 `source:destination` 쌍을 지정합니다(예: `"class-a:class-b,class-c:class-d"`). `*:destination-class`를 와일드카드로 사용하여 매핑되지 않은 모든 PVC를 단일 대상 스토리지 클래스에 매핑합니다. `--storageclass-mapping`가 설정된 경우 `--storage-class`는 무시됩니다.

단일 대상 스토리지 클래스를 사용하는 예시 (**--storage-class**):

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (선택 사항) 대상 클러스터에서 복제 관계의 상태 및 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

대상 클러스터로 페일오버합니다

Trident Protect를 사용하면 복제된 애플리케이션을 대상 클러스터로 장애 조치할 수 있습니다. 이 절차는 복제 관계를 중지하고 대상 클러스터에서 앱을 온라인으로 전환합니다. Trident Protect는 소스 클러스터에서 앱이 작동 중이면 앱을 중지하지 않습니다.

이 절차는 최신 스냅샷으로 페일오버합니다. 이전 스냅샷으로 페일오버하려면 "Trident Protect를 사용하여 이전 스냅샷으로 페일오버합니다."을(를) 참조하십시오.

단계

1. (선택 사항) KubeVirt VM의 MAC 주소를 유지하려면 승격하기 전에 AppMirrorRelationship에 주석을 추가하십시오. "고급 Trident Protect 복원 설정 사용"를 참조하십시오.

```
kubectl annotate appmirrorrelationship <relationship_name> -n
<application_namespace> protect.trident.netapp.io/retain-mac-address
="true"
```

2. 대상 클러스터에서 AppMirrorRelationship CR 파일(예 `trident-protect-relationship.yaml`:)을 편집하고 * spec.desiredState* 값을 로 변경합니다. `Promoted`
3. CR 파일을 저장합니다.
4. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. (선택 사항) 페일오버된 애플리케이션에 필요한 보호 스케줄을 생성합니다.

6. (선택 사항) 복제 관계의 상태 및 상태를 확인합니다.

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

페일오버된 복제 관계를 다시 동기화합니다

재동기화 작업은 복제 관계를 다시 설정합니다. 재동기화 작업을 수행하면 원래 소스 애플리케이션이 실행 중인 애플리케이션이 되고 대상 클러스터에서 실행 중인 애플리케이션에 대한 변경 내용은 모두 삭제됩니다.

이 프로세스는 복제를 다시 설정하기 전에 대상 클러스터에서 앱을 중지합니다.

중요함	페일오버 중에 대상 애플리케이션에 기록된 모든 데이터가 손실됩니다.

단계

1. 선택 사항: 소스 클러스터에서 소스 애플리케이션의 스냅샷을 생성합니다. 이렇게 하면 소스 클러스터의 최신 변경 사항이 캡처됩니다.
2. 대상 클러스터에서 AppMirrorRelationship CR 파일(예 `trident-protect-relationship.yaml`:)을 편집하고 spec.desiredState 값을 로 변경합니다. `Established`
3. CR 파일을 저장합니다.
4. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. 대상 클러스터에서 페일오버된 애플리케이션을 보호하기 위해 보호 스케줄을 생성한 경우 제거하십시오. 남아 있는 스케줄은 볼륨 스냅숏에 장애를 일으킵니다.

페일오버된 복제 관계를 역방향으로 재동기화합니다

페일오버된 복제 관계를 역동기화하는 경우 대상 애플리케이션은 소스 애플리케이션이 되고 소스는 대상이 됩니다. 페일오버 중에 대상 애플리케이션에 대한 변경 사항은 유지됩니다.

단계

1. 원래 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다. 그러면 대상이 원본이 됩니다. 새 대상 클러스터에 남아 있는 보호 스케줄이 있는 경우 제거합니다.
2. 관계를 설정할 때 원래 사용했던 CR 파일을 반대 클러스터에 적용하여 복제 관계를 설정합니다.
3. 새 대상(원래 소스 클러스터)이 두 AppVault CRS로 구성되어 있는지 확인합니다.
4. 반대 클러스터에서 복제 관계를 설정하여 반대 방향에 대한 값을 구성합니다.

애플리케이션 복제 방향을 반대로 전환합니다

복제 방향을 반대로 하면 Trident Protect는 원래 소스 스토리지 백엔드로 복제를 계속 진행하면서 애플리케이션을 대상 스토리지 백엔드로 이동합니다. Trident Protect는 소스 애플리케이션을 중지하고 대상 앱으로 장애 조치하기 전에 대상에 데이터를 복제합니다.

이 경우 소스와 대상을 스와핑합니다.

단계

1. 소스 클러스터에서 종료 스냅샷을 생성합니다.

CR을 사용하여 종료 스냅샷을 생성합니다

a. 소스 애플리케이션에 대한 보호 정책 일정을 해제합니다.

b. ShutdownSnapshot CR 파일 생성:

i. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-shutdownsnapshot.yaml`).

ii. 다음 특성을 구성합니다.

- **metadata.name**: (*required*) 사용자 정의 리소스의 이름입니다.
- * spec.AppVaultRef *: (*required*) 이 값은 원본 응용 프로그램에 대한 AppVault의 metadata.name 필드와 일치해야 합니다.
- **spec.ApplicationRef**: (*required*) 이 값은 소스 응용 프로그램 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. 파일을 올바른 값으로 채운 후 `trident-protect-shutdownsnapshot.yaml` CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-namespace
```

CLI를 사용하여 종료 스냅샷을 생성합니다

a. 괄호 안의 값을 사용자 환경의 정보로 대체하여 종료 스냅샷을 만듭니다. 예를 들면 다음과 같습니다.

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot> --appvault <my_vault> --app <app_to_snapshot> -n
```

```
<application_namespace>
```

2. 소스 클러스터에서 종료 스냅샷이 완료된 후 종료 스냅샷의 상태를 가져옵니다.

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. 소스 클러스터에서 다음 명령을 사용하여 * shutdownsnapshot.status.appArchivePath * 의 값을 찾고 파일 경로의 마지막 부분(basename라고도 함. 이것은 마지막 슬래시 다음에 모두 있음)을 기록합니다.

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. 다음 변경 사항을 적용하여 새 대상 클러스터에서 새 소스 클러스터로 페일오버를 수행합니다.

참고	페일오버 절차의 2단계에서 AppMirrorRelationship CR 파일에 필드를 포함하고 `spec.promotedSnapshot` 위의 3단계에서 기록한 기본 이름으로 값을 설정합니다.

5. 의 역동기화 단계를 페일오버된 복제 관계를 역방향으로 재동기화합니다수행합니다.
6. 새 소스 클러스터에서 보호 스케줄을 설정합니다.

결과

역방향 복제 때문에 다음 작업이 발생합니다.

- 원본 소스 앱의 Kubernetes 리소스에 대한 스냅샷이 생성됩니다.
- 앱의 Kubernetes 리소스를 삭제하여 원본 소스 앱의 Pod를 정상적으로 중지할 수 있습니다(PVC 및 PVS를 그대로 둡니다).
- 포드가 종료된 후 앱 볼륨의 스냅샷이 촬영되고 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨이 읽기/쓰기 준비가 되었습니다.
- 앱의 Kubernetes 리소스는 원래 소스 애플리케이션이 종료된 후 복제된 볼륨 데이터를 사용하여 사전 종료 스냅샷에서 복구됩니다.
- 복제는 반대 방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백합니다

Trident Protect를 사용하면 다음과 같은 작업 순서를 통해 장애 조치 작업 후에 "장애 복구"를 달성할 수 있습니다. 원래 복제 방향을 복원하기 위한 이 워크플로에서 Trident Protect는 복제 방향을 반전하기 전에 모든 애플리케이션 변경 사항을 원래 소스 애플리케이션으로 복제(재동기화)합니다.

이 프로세스는 대상에 대한 페일오버를 완료한 관계로부터 시작되며 다음 단계를 포함합니다.

- 페일오버된 상태로 시작합니다.

- 복제 관계를 역방향으로 다시 동기화합니다.

주의	일반 재동기화 작업을 수행하지 마십시오. 그러면 페일오버 절차 중에 대상 클러스터에 기록된 데이터가 삭제됩니다.

- 복제 방향을 반대로 바꿉니다.

단계

1. 페일오버된 복제 관계를 역방향으로 재동기화합니다다음 단계를 수행합니다.
2. 애플리케이션 복제 방향을 반대로 전환합니다다음 단계를 수행합니다.

복제 관계를 삭제합니다

언제든지 복제 관계를 삭제할 수 있습니다. 애플리케이션 복제 관계를 삭제하면 서로 관계가 없는 두 개의 개별 애플리케이션이 생성됩니다.

단계

1. 현재 대상 클러스터에서 AppMirrorRelationship CR을 삭제합니다.

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Trident Protect를 사용하여 이전 스냅샷으로 페일오버합니다**.**

Trident Protect를 사용하면 복제된 애플리케이션을 이전 스냅샷으로 페일오버하여 데이터 손상 또는 유출 이전 시점으로 복구할 수 있습니다. 기본적으로 페일오버는 가장 최근에 전송된 스냅샷을 사용합니다.

주의	이전 스냅샷으로 페일오버하면 ONTAP SnapMirror가 타겟 클러스터에서 선택한 스냅샷보다 최신인 스냅샷을 삭제합니다. 이는 해당 스냅샷이 더 이상 유효하지 않기 때문입니다. 이러한 데이터 손실이 허용되는 경우에만 이 절차에 주석을 추가하십시오.

시작하기 전에

- AppMirrorRelationship이(가) 존재하며 `Established` 상태입니다. 이를 생성하려면 "복제 관계를 설정합니다"을(를) 참조하십시오. 상태를 확인하려면 `kubectl get amr -n <namespace> <relationship_name> -o=jsonpath='{.status}'`을(를) 실행하십시오.
- 소스 및 타겟 클러스터는 SnapMirror 복제를 사용하는 ONTAP 스토리지 백엔드를 사용합니다. Azure NetApp Files 및 기타 비 ONTAP 백엔드는 지원되지 않습니다. "SnapMirror 복제에 대한 요구 사항"을(를) 참조하십시오.
- 소스 애플리케이션에 스냅샷 일정(`snapshotRetention` 1보다 큼)이 있습니다.

스냅샷을 식별하십시오

각 스냅샷에는 완료 시간이 있습니다. 소스 클러스터에서 `kubectl`을(를) 사용하거나 CLI를 사용하여 해당 아카이브 경로를 확인하십시오.

참고	소스 클러스터가 다운되었거나 스냅샷 사용자 지정 리소스를 사용할 수 없는 경우 CLI를 사용하십시오. 타겟 클러스터 또는 소스 AppVault가 있는 모든 클러스터에서 명령을 실행하십시오.

kubectl을 사용하십시오

소스 클러스터에서 `kubectl`을(를) 사용하여 아카이브 경로를 가져오십시오.

1. 소스 클러스터에서 스냅샷 목록을 확인하고 원하는 스냅샷의 `completionTimestamp`을(를) 기록하십시오:

```
kubectl get snapshots -n my-app-namespace
```

2. 해당 아카이브 경로를 가져오십시오:

```
kubectl get snapshots <snapshot_name> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

CLI를 사용합니다

Trident Protect CLI를 사용하여 아카이브 경로를 가져옵니다. 타겟 클러스터 또는 소스 AppVault가 있는 모든 클러스터에서 스냅샷 목록을 확인합니다. `TIMESTAMP`는 완료 시간이고 `PATH`는 아카이브 경로입니다.

```
tridentctl-protect get appvaultcontent <source_appvault_name> \
--app <app_name> \
--show-resources snapshot \
--show-paths
```

참고 | 소스 애플리케이션의 AppVault CR은 명령을 실행하는 클러스터에 존재해야 하며, 사용 환경은 AppVault CR에 정의된 오브젝트 스토리지 버킷에 연결할 수 있어야 합니다. 네트워크 제한으로 인해 액세스할 수 없는 경우 클러스터의 Pod 내에서 CLI를 실행하십시오.

경로에서 마지막 슬래시 뒤에 오는 모든 부분이 기본 이름입니다. 예를 들면 다음과 같습니다.

```
20231114173500_schedule-20231114173500_f8df8bd3-a727-4a14-b954-
9666aa618b55
```

참고 | 스냅샷 사용자 지정 리소스의 이름이 아닌 기본 이름을 사용하십시오.

결과

선택한 스냅샷의 기본 이름이 기록됩니다.

선택한 스냅샷으로 페일오버합니다

승격할 스냅샷을 지정하고, 주석을 추가하여 데이터 손실을 확인하십시오.

단계

1. 타겟 클러스터에서 AppMirrorRelationship에 어노테이션을 추가합니다. 어노테이션이 없으면 장애 조치 시 가장

최근에 전송된 스냅샷만 사용할 수 있습니다.

```
kubectl annotate appmirrorrelationship <relationship_name> -n my-app-
namespace protect.trident.netapp.io/allow-promotion-with-older-
snapshots="true"
```

2. AppMirrorRelationship CR 파일을 편집합니다(예: `trident-protect-relationship.yaml`):
   - *spec.desiredState*를 `Promoted`로 설정합니다.
   - *spec.promotedSnapshot*을 스냅샷을 식별하십시오에 기록한 기본 이름으로 설정하십시오.

   예를 들면 다음과 같습니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: <relationship_name>
  namespace: <namespace>
  annotations:
    protect.trident.netapp.io/allow-promotion-with-older-snapshots:
"true"
spec:
  desiredState: Promoted
  promotedSnapshot: <basename>
```

3. CR 적용:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. 관계 상태를 확인하십시오:

```
kubectl get amr -n my-app-namespace <relationship_name> -o=jsonpath
='{.status}' | jq
```

결과

관계가 `Promoted` 상태로 이동하고, 애플리케이션은 선택된 스냅샷의 데이터를 사용하여 타겟 클러스터에서 실행됩니다.

관계가 `Promoting` 상태로 유지되고 `status.error`에 `The snapshot provided is invalid or older than the most recently transferred snapshot`이(가) 표시되면 어노테이션이 누락되었거나 `true`(으)로 설정되지 않은 것입니다. 어노테이션을 추가하고 다시 시도하십시오.

다음 단계

원래 클러스터로 돌아가려면 "역방향 재동기화"을(를) 수행한 다음 "복제 방향을 반전하십시오".

주의	원래 클러스터로 돌아가려면 일반 재동기화를 수행하지 마십시오. 일반 재동기화는 장애 조치된 애플리케이션을 삭제하고 관계를 최신 소스 스냅샷으로 되돌립니다.

관련 정보

- "Trident Protect를 사용하여 애플리케이션 복제를 설정하고 관리합니다."
- "대상 클러스터로 페일오버합니다"
- "페일오버된 복제 관계를 다시 동기화합니다"
- "애플리케이션을 원래 소스 클러스터로 페일백합니다"
- "데이터 보호 스케줄을 생성합니다"

Trident Protect를 사용하여 애플리케이션 마이그레이션

백업 데이터를 복원하여 클러스터 간이나 다른 스토리지 클래스로 애플리케이션을 마이그레이션할 수 있습니다.

참고	애플리케이션을 마이그레이션하면 애플리케이션에 대해 구성된 모든 실행 후크가 앱과 함께 마이그레이션됩니다. 복원 후 실행 후크가 있는 경우 복구 작업의 일부로 자동으로 실행됩니다.

백업 및 복원 작업

다음 시나리오에 대한 백업 및 복원 작업을 수행하려면 특정 백업 및 복원 작업을 자동화할 수 있습니다.

같은 클러스터에 클론 복제

애플리케이션을 동일한 클러스터에 복제하려면 스냅샷 또는 백업을 생성하고 데이터를 동일한 클러스터에 복구합니다.

단계

1. 다음 중 하나를 수행합니다.
   a. "스냅샷을 생성합니다"..
   b. "백업을 생성합니다"..
2. 동일한 클러스터에서 스냅샷 또는 백업을 생성했는지에 따라 다음 중 하나를 수행합니다.
   a. "스냅샷에서 데이터를 복원합니다"..
   b. "백업에서 데이터를 복원합니다"..

다른 클러스터에 클론 복제

다른 클러스터에 애플리케이션을 복제하려면(클러스터 간 복제 수행) 소스 클러스터에 백업을 만든 다음, 해당 백업을 다른 클러스터로 복원합니다. 대상 클러스터에 Trident Protect가 설치되어 있는지 확인하세요.

참고	을 사용하여 서로 다른 클러스터 간에 애플리케이션을 복제할 수 "SnapMirror 복제"있습니다.

단계

1. "백업을 생성합니다"..
2. 백업이 포함된 객체 스토리지 버킷의 AppVault CR이 대상 클러스터에 구성되어 있는지 확인합니다.
3. 대상 클러스터에서"백업에서 데이터를 복원합니다".

한 스토리지 클래스에서 다른 스토리지 클래스로 애플리케이션을 마이그레이션합니다

대상 스토리지 클래스에 백업을 복원하여 애플리케이션을 한 스토리지 클래스에서 다른 스토리지 클래스로 마이그레이션할 수 있습니다.

예를 들어 복원 CR에서 비밀 제외):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

CR을 사용하여 스냅샷을 복원합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-snapshot-restore-cr.yaml`지정합니다.
2. 생성한 파일에서 다음 속성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.appArchivePath *: 스냅샷 내용이 저장되는 AppVault 내부 경로입니다. 다음 명령을 사용하여 이 경로를 찾을 수 있습니다.

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o
     jsonpath='{.status.appArchivePath}'
     ```

- * spec.appVaultRef *: (*required*) 스냅샷 내용이 저장된 AppVault의 이름입니다.
- **spec.namespaceMapping**: 복구 작업의 소스 네임스페이스를 대상 네임스페이스에 매핑하는 것입니다. `my-source-namespace`및 를 `my-destination-namespace` 사용자 환경의 정보로 바꿉니다.

```
---
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. 복원할 응용 프로그램의 특정 리소스만 선택해야 하는 경우 특정 레이블로 표시된 리소스를 포함하거나 제외하는 필터링을 추가합니다.
   - **resourceFilter.resourceSelectionCriteria**:(필터링에 필요) `include or exclude` resourceMatchers에 정의된 리소스를 포함하거나 제외하는 데 사용합니다. 다음 resourceMatchers 매개 변수를 추가하여 포함하거나 제외할 리소스를 정의합니다.
     - **resourceFilter.resourceMatchers**: resourceMatcher 개체의 배열입니다. 이 배열에서 여러 요소를 정의하는 경우 해당 요소는 OR 연산으로 일치하고 각 요소(그룹, 종류, 버전) 내의 필드는 AND 연산으로 일치합니다.
       - **resourceMatchers[].group**:(*Optional*) 필터링할 리소스의 그룹입니다.
       - * resourceMatchers [].kind *: (*Optional*) 필터링할 리소스의 종류입니다.
       - **resourceMatchers [ ].version**: (*Optional*) 필터링할 리소스의 버전입니다.
       - **resourceMatchers[].names**:(*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 이름입니다.
       - *resourceMatchers [].namespaces *: (*Optional*) 필터링할 리소스의 Kubernetes metadata.name 필드에 있는 네임스페이스입니다.
       - **resourceMatchers [ ].labelSelectors**: (*Optional*) 에 정의된 대로 리소스의 Kubernetes metadata.name 필드에 있는 레이블 선택기 문자열입니다. "Kubernetes 문서" 예를 들면 다음과 `"'trident.netapp.io/os=linux"`같습니다.

         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. 파일을 올바른 값으로 채운 후 `trident-protect-snapshot-restore-cr.yaml` CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

CLI를 사용하여 스냅샷을 복원합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 스냅샷을 다른 네임스페이스로 복원합니다.

   - `snapshot`인수에 네임스페이스 및 스냅숏 이름이 형식으로 `<namespace>/<name>` 사용됩니다.
   - 이 `namespace-mapping` 인수는 콜론으로 구분된 네임스페이스를 사용하여 소스 네임스페이스를 올바른 대상 네임스페이스에 형식 `source1:dest1,source2:dest2`으로 매핑합니다.

     예를 들면 다음과 같습니다.

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

KubeVirt 가상 머신을 Trident Protect로 보호합니다

Trident Protect를 사용하면 OpenShift Virtualization에서 실행되는 KubeVirt 가상 머신(VM)을 백업하고 복원할 수 있습니다. 하나 이상의 네임스페이스에 있는 모든 VM을 한 번에 보호하거나 `includedVirtualMachines` 필드를 사용하여 이름별로 특정 VM을 지정할 수 있습니다. Trident Protect는 백업 중에 모든 종속 VM 리소스를 자동으로 수집하며, 복원 중에는 전체 아카이브를 복구하거나 개별 VM을 선택적으로 복원할 수 있습니다.

KubeVirt 보호 모드

Trident Protect는 보호할 KubeVirt VM을 정의하는 두 가지 방법을 지원합니다. 애플리케이션을 정의할 때 이 중 하나의 모드를 선택하십시오.

모드	필드에 입력합니다	설명
네임스페이스 기반	`includedNamespaces`	하나 이상의 네임스페이스에서 일치하는 모든 리소스를 보호합니다. 네임스페이스 항목에 `labelSelector`를 사용하여 동적으로 포함될 VM을 제어할 수 있습니다.
VM 범위	`includedVirtualMachines`	네임스페이스 및 이름을 기준으로 특정 KubeVirt VM을 보호합니다. VM이 여러 네임스페이스에 분산되어 있거나 모든 네임스페이스 리소스를 보호할 필요가 없는 경우에 이 모드를 사용하십시오.

반드시 정확히 하나의 필드 (`includedNamespaces` 또는 `includedVirtualMachines`)를 `ApplicationSpec`에 지정해야 합니다. 동일한 애플리케이션에서 두 필드를 모두 사용할 수 없습니다.

두 모드 모두 추가 필터링을 지원합니다.

- `resourceFilter`을(를) 사용하여 특정 리소스를 포함하거나 제외합니다.
- `includedClusterScopedResources`을(를) 사용하여 `StorageClass` 객체와 같은 호환 가능한 클러스터 범위 리소스를 포함합니다.

참고

- Trident Protect는 데이터 보호 작업 중에 KubeVirt VM 파일 시스템을 동결 및 해제하여 일관성을 보장할 수 있습니다. 구성 세부 정보는 "KubeVirt VM으로 데이터 보호"을 참조하십시오.
- 가상 머신의 모든 디스크를 동일한 시점에 스냅샷하려면 볼륨 그룹 스냅샷을 사용하십시오. "Trident Protect에서 볼륨 그룹 스냅샷 사용"을(를) 참조하십시오.

백업 및 스냅샷 중에 수집된 리소스

KubeVirt VM이 포함된 애플리케이션의 백업 또는 스냅샷을 생성할 때 Trident Protect는 선택한 VM 리소스를 수집하고 해당 종속 리소스를 자동으로 검색합니다.

수집 동작은 다음 두 가지 경우에 적용됩니다.

- VM 범위 애플리케이션은 `includedVirtualMachines`을(를) 사용합니다.
- KubeVirt `VirtualMachine` 리소스를 포함하는 네임스페이스 기반 애플리케이션.

VM 범위 컬렉션

VM 범위 애플리케이션의 경우, Trident Protect는 `includedVirtualMachines`에 나열된 각 `VirtualMachine`에서 시작하여 해당 VM을 복원하는 데 필요한 종속 리소스를 검색합니다.

리소스 유형	범위	검색 방법
`VirtualMachine`	네임스페이스	`includedVirtualMachines` 목록에서 직접 일치하는 항목입니다.
`VirtualMachineInstance`	네임스페이스	VM이 실행 중인 경우 VM과 동일한 네임스페이스 및 이름을 사용합니다.
`VirtualMachineInstancetype`	네임스페이스	`spec.instancetype`에 의해 참조됩니다 `kind: VirtualMachineInstancetype`.

리소스 유형	범위	검색 방법
`VirtualMachineClusterInstancetype`	클러스터	`spec.instancetype`에 의해 참조되며, `kind`가 설정되지 않았거나 `kind: VirtualMachineClusterInstancetype`인 경우에 해당합니다.
`VirtualMachinePreference`	네임스페이스	`spec.preference`에 의해 참조됨 `kind: VirtualMachinePreference`.
`VirtualMachineClusterPreference`	클러스터	`spec.preference`에 의해 참조되며 `kind`가 설정되지 않았거나 `kind: VirtualMachineClusterPreference`인 경우에 사용됩니다.
`DataVolume`	네임스페이스	참조 대상: `spec.template.spec.volumes[].dataVolume.name`.
`PersistentVolumeClaim`	네임스페이스	DataVolume과 동일한 이름을 사용하는 DataVolume 지원 PVC에서 검색되었으며, `persistentVolumeClaim.claimName`의 직접 PVC 참조, `memoryDump.claimName`의 메모리 덤프 PVC 및 `ephemeral.persistentVolumeClaim.claimName`의 임시 PVC가 포함됩니다.
`PersistentVolume`	클러스터	수집된 각 PVC의 백업 PV에서 `spec.volumeName`을(를) 사용하여 검색됩니다.
`Secret`	네임스페이스	볼륨 비밀에서 검색됨 `secret.secretName`; containerDisk 이미지 풀 시크릿; CloudInitNoCloud 및 CloudInitConfigDrive 사용자 데이터 및 네트워크 데이터 시크릿 참조; Sysprep 시크릿; KernelBoot 컨테이너 이미지 풀 시크릿; 및 AccessCredentials SSH 또는 암호 시크릿.
`ConfigMap`	네임스페이스	볼륨 ConfigMaps `configMap.name` 및 Sysprep ConfigMaps에서 검색되었습니다.
`ServiceAccount`	네임스페이스	`serviceAccount.serviceAccountName` 볼륨 참조에서 검색되었습니다.
`NetworkAttachmentDefinition`	네임스페이스	Multus 네트워크에서 `spec.template.spec.networks[].multus.networkName` 네트워크 이름에 `/`이(가) 포함되지 않은 경우 검색됩니다. 동일 네임스페이스의 NetworkAttachmentDefinition 참조만 수집됩니다. 다른 네임스페이스 참조는 건너뜁니다.
`Pod`	네임스페이스	`virt-launcher`레이블 선택기 `kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`를 사용하여 Pods에 대해 검색되었습니다.
스토리지 클래스, Trident 볼륨 및 Trident 백엔드	클러스터	VM에 대해 검색된 PVC를 기반으로 네임스페이스 기반 애플리케이션과 동일한 방식으로 수집됩니다.

참고 VM 범위의 네임스페이스 리소스 수집이 완료되면 Trident Protect는 표준 `resourceFilter`을 적용합니다(구성된 경우). 이후 클러스터 범위 리소스 수집은 정상적으로 진행됩니다.

네임스페이스 기반 컬렉션

네임스페이스 기반 애플리케이션의 경우, Trident Protect는 선택한 네임스페이스를 스캔하고 일치하는 모든 리소스를

수집합니다. KubeVirt `VirtualMachine` 객체가 발견되면 Trident Protect는 VM 범위 애플리케이션과 마찬가지로 인스턴스 유형, 환경 설정 및 `virt-launcher` Pod를 포함한 해당 종속 리소스를 자동으로 수집합니다.

즉, 애플리케이션이 `includedNamespaces`로 정의된 경우에도 VM별 리소스가 백업 또는 스냅샷에 포함됩니다.

네임스페이스 기반 애플리케이션 정의를 사용하여 **VM** 보호

네임스페이스 기반 애플리케이션을 생성하면 Trident Protect는 선택한 네임스페이스를 스캔하여 KubeVirt VM을 포함한 모든 리소스를 수집합니다. Trident Protect는 발견된 각 가상 머신에 대해 해당 VM을 백업하고 복원하는 데 필요한 종속 리소스도 수집합니다.

예시 네임스페이스 기반 애플리케이션 CR:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

애플리케이션을 생성한 후에는 표준 Trident Protect 워크플로를 사용하여 백업 및 스냅샷을 생성하십시오.

- 스냅샷 YAML 예제는 "필요 시 스냅샷을 생성합니다"을 참조하십시오.
- 백업 YAML 예제는 "필요 시 백업을 생성합니다"을 참조하십시오.

네임스페이스 기반 **VM** 애플리케이션의 동작 복원

네임스페이스 기반 애플리케이션은 선택한 네임스페이스에서 리소스를 수집하므로 백업 또는 스냅샷에 KubeVirt 런타임 객체 및 KubeVirt에서 관리하는 메타데이터가 포함될 수 있습니다. 이러한 객체 중 일부는 소스 클러스터의 런타임 상태를 나타내므로 직접 복원해서는 안 됩니다.

복원 과정에서 Trident Protect는 KubeVirt를 인식하는 변환을 적용하여 소스 클러스터 메타데이터를 제거하고 임시 KubeVirt 리소스를 건너뜁니다. 이를 통해 KubeVirt가 복원된 환경에서 VM을 깔끔하게 다시 생성할 수 있습니다.

리소스	변환
VirtualMachine	`kubevirt.io/`으로 시작하는 레이블과 주석을 제거합니다.
VirtualMachineInstance	`kubevirt.io/`으로 시작하는 레이블과 주석을 제거합니다.
VirtualMachineInstanceMigration	무시되었고 복구되지 않았습니다.
VirtualMachineSnapshot	무시되었고 복구되지 않았습니다.
VirtualMachineSnapshotContent	무시되었고 복구되지 않았습니다.
Pod (`virt-launcher`)	포드에 `kubevirt.io=virt-launcher` 라벨이 있는 경우 건너뜁니다.

참고

- Trident Protect는 종속성 문제를 방지하기 위해 정의된 순서대로 KubeVirt 리소스를 복원합니다.

CDI DataVolumes (`cdi.kubevirt.io`)가 먼저 복원되어 VM 객체가 생성되기 전에 VM 디스크 데이터를 사용할 수 있게 됩니다. 그 후 VirtualMachines (`kubevirt.io`)가 VirtualMachineInstances 이전에 복원됩니다.

- 기본적으로 Trident Protect는 복원된 각 VM의 첫 번째 네트워크 인터페이스에서 MAC 주소를 제거합니다. 이를 유지하려면 `protect.trident.netapp.io/retain-mac-address` 주석을 사용하십시오. "고급 Trident Protect 복원 설정 사용"을(를) 참조하십시오.

레이블 선택기를 사용한 동적 **VM** 보호

각 VM 이름을 나열하지 않고 VM을 동적으로 보호하려면 레이블 선택기를 사용하는 네임스페이스 기반 애플리케이션 정의를 사용하십시오.

VirtualMachine CR에만 레이블이 필요합니다. DataVolumes, PVC, Secrets, ConfigMaps와 같은 종속 리소스는 자동으로 수집됩니다.

단계

1. 가상 머신 CR에 공유 레이블을 지정합니다.

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. `includedNamespaces` 및 `labelSelector`을(를) 사용하여 애플리케이션 CR을 생성합니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. Trident Protect의 표준 워크플로를 사용하여 백업 및 복원을 생성합니다.

추가 정보:

- 레이블을 변경하면 언제든지 VM을 보호 대상에 추가하거나 제거할 수 있습니다.

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
```

```
kubectl label virtualmachine <vm_name_to_remove> app
=<different_label_value> -n <vm_namespace>
```

- 레이블 선택기를 여러 네임스페이스에 걸쳐 사용할 수도 있습니다.

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

includedVirtualMachines로 특정 **VM** 보호

하나 이상의 네임스페이스에 걸쳐 지정된 VirtualMachines 집합을 보호하고 아카이브에서 특정 VM을 선택적으로 복원하려는 경우 `includedVirtualMachines`를 사용합니다.

응용 프로그램 및 복원 CR 모두 동일한 `includedVirtualMachines` 선택기 형식을 사용합니다.

이 선택자를 다양한 상황에서 사용하세요:

- 애플리케이션 CR에서는 어떤 VM을 보호할지 정의합니다.
- 복원 CR에서는 백업 또는 스냅샷 아카이브에서 복원할 VM을 정의합니다.

참고	CR 복원 시 `includedVirtualMachines`는 선택 사항입니다. 이 필드를 포함하지 않으면 전체 아카이브가 복원됩니다. 지정하면 나열된 VM과 해당 종속 리소스만 복원됩니다.

아래 YAML은 공유 선택자 구조를 보여줍니다.

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

참고	`includedVirtualMachines`의 여러 항목이 서로 다른 네임스페이스를 참조할 수 있으므로 다중 네임스페이스를 지원하는 VM 인식 애플리케이션 정의 및 복원이 가능합니다.

필드에 입력합니다	유형	설명
네임스페이스	문자열	대상 가상 머신 객체가 포함된 Kubernetes 네임스페이스입니다.
이름	문자열 목록	해당 네임스페이스에 있는 하나 이상의 가상 머신 이름입니다.

가상 머신 기반 애플리케이션 예시:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

특정 KubeVirt VM 복원

복원 CR에서 `includedVirtualMachines`을 사용하면 백업, 스냅샷 또는 복제된 스냅샷 아카이브에서 선택한 KubeVirt VM만 복원할 수 있습니다.

복원 CR에 `includedVirtualMachines`이 지정된 경우 Trident Protect는 나열된 VirtualMachines과 해당 종속 리소스(예: `VirtualMachineInstance`, DataVolume, PersistentVolumeClaim, PersistentVolume, Secret, ConfigMap, ServiceAccount, 기타 VM 관련 개체)만 복원합니다.

복원 CR에서 `includedVirtualMachines`을(를) 지정하지 않으면 Trident Protect는 복원 CR 구성에 따라 전체 아카이브를 복원합니다.

지원되는 복원 CR 및 CLI 명령

다음 표는 지원되는 복원 CR `includedVirtualMachines`과 해당 CLI 명령을 나열합니다.

CR 복원	CLI 명령	설명
BackupRestore	tridentctl-protect create backuprestore	새롭거나 다른 네임스페이스 매핑으로 백업에서 복원합니다. namespaceMapping 및 `storageClassMapping`을(를) 지원합니다.

CR 복원	CLI 명령	설명
BackupInplaceRestore	tridentctl-protect create backupinplacerestore	백업에서 원래 네임스페이스 또는 여러 네임스페이스로 복원합니다.
SnapshotRestore	tridentctl-protect create snapshotrestore	스냅샷에서 새롭거나 다른 네임스페이스 매핑으로 복원합니다. namespaceMapping 및 `storageClassMapping`를 지원합니다.
SnapshotInplaceRestore	tridentctl-protect create snapshotinplacerestore	스냅샷에서 원래 네임스페이스 또는 네임스페이스들로 복원합니다.
ReplicateSnapshotRestore	YAML만 사용하거나, 환경에서 지원되는 CLI 명령을 사용하십시오(사용 가능한 경우).	복제된 스냅샷에서 복원합니다. namespaceMapping, storageClassMapping, 및 inPlaceRestore 플래그를 사용한 제자리 복원을 지원합니다.

참고

tridentctl-protect CLI에는 복원 명령에 includedVirtualMachines`에 대한 플래그가 없습니다. 특정 VM을 복원하려면 `kubectl apply`을(를) 사용하여 복원 CR YAML 파일을 직접 적용해야 합니다. 처음부터 YAML을 작성하거나, `--dry-run --output yaml`와(과) 함께 CLI를 사용하여 시작 파일을 생성하고, `spec 아래에 includedVirtualMachines 필드를 추가한 다음 이를 적용할 수 있습니다.

VM 범위 복원 동작

복원 CR의 includedVirtualMachines 필드는 애플리케이션 CR과 동일한 선택기 형식을 사용합니다.

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

CR 복원의 경우 `includedVirtualMachines`의 네임스페이스 값은 백업, 스냅샷 또는 복제된 스냅샷 아카이브에 존재하는 소스 네임스페이스를 참조합니다.

`namespaceMapping`를 사용하는 경우, Trident Protect는 먼저 소스 아카이브에서 요청된 VM을 선택한 다음 복원 중에 네임스페이스 매핑을 적용합니다. `storageClassMapping`를 사용하는 경우, Trident Protect는 복원 변환 단계에서 스토리지 클래스 매핑을 적용합니다.

CLI를 사용하여 복원 YAML 생성

`tridentctl-protect`을 사용하여 기본 복원 CR을 생성한 다음 YAML을 편집하여

`includedVirtualMachines`을 추가할 수 있습니다.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

생성된 YAML을 편집하고 `includedVirtualMachines` 필드를 `spec` 아래에 추가한 다음 파일을 적용하세요.

```
kubectl apply -f <restore_name>.yaml
```

예시: **VM** 기반 애플리케이션 정의

다음 애플리케이션은 두 개의 네임스페이스에 걸쳐 있는 세 개의 특정 가상 머신을 보호합니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

예시: 백업 생성

VM 기반 애플리케이션의 백업을 생성할 때는 표준 백업 CR을 사용합니다. 백업 CR에는 VM별 필드를 입력할 필요가 없습니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
  namespace: prod
```

```
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

예시: 백업에서 단일 가상 머신 복원

다음 예제는 백업 아카이브에서 `database-server`만 복원합니다. 아카이브에 있는 다른 VM(예: `app-server` 및 `prometheus-vm`)은 건너뜁니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

Only `database-server` 및 DataVolumes, PVCs, PVs, Secrets, ConfigMaps, 관련 KubeVirt 리소스와 같은 종속 리소스만 복원됩니다.

예시: 백업에서 단일 가상 머신을 제자리에 복원

다음 예제는 `database-server`을(를) 백업 아카이브에서 원래 네임스페이스로만 복원합니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
```

```
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

예시: 스냅샷에서 단일 가상 머신 복원

다음 예제는 스냅샷 아카이브에서 `ubuntu-vm-blue-a627be38`만 복원합니다.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

VM 복원 기능을 다른 필터와 결합

`includedVirtualMachines`을 동일한 복원 CR에서 `resourceFilter`, `namespaceMapping`, `storageClassMapping`와 함께 사용하여 복원 대상을 세밀하게 조정할 수 있습니다.

를 추가하면 `resourceFilter`, Trident Protect는 먼저 `includedVirtualMachines`에 나열된 VM과 해당 VM에 종속된 리소스를 선택합니다. 그런 다음 리소스 필터를 적용하여 결과에 포함되거나 제외될 항목을 더욱 세분화합니다.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
```

```
      - kinds: ["PersistentVolumeClaim"]
        names: ["<pvc_name>"]
```

`BackupRestore`, `SnapshotRestore` 및 `ReplicateSnapshotRestore`의 경우, `namespaceMapping` 및 `storageClassMapping`는 `includedVirtualMachines`와 함께 작동합니다. `includedVirtualMachines`의 네임스페이스 참조는 아카이브에 나타나는 소스 네임스페이스에 해당합니다. 네임스페이스 및 스토리지 클래스 매핑은 필터링이 완료된 후 복원 변환 단계에서 적용됩니다.

복원 동작에 대한 자세한 내용은 "Trident Protect를 사용하여 애플리케이션 복원"을 참조하십시오.

CLI를 사용하여 **VM** 범위 애플리케이션 생성

이 `--virtual-machines` 플래그를 사용하면 전체 네임스페이스 대신 특정 KubeVirt VM에 애플리케이션 범위를 지정할 수 있습니다. 이 `--virtual-machines` 플래그는 `--namespaces`와 함께 사용할 수 없습니다.

단계

1. 다음 명령 중 하나를 실행하여 애플리케이션을 생성하십시오. 대괄호 안의 값은 사용자 환경 정보로 바꿔야 합니다. `--virtual-machines` 플래그의 형식은 `<namespace>(<vm_name>)`입니다. 여러 개의 가상 머신(VM)을 사용하는 경우 괄호 안에 쉼표로 구분된 이름을 사용하십시오.
   - 네임스페이스 내 단일 VM:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - 여러 네임스페이스에 걸쳐 있는 여러 VM:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

관련 정보

- "관리할 애플리케이션을 정의합니다"
- "애플리케이션 보호"
- "Trident Protect를 사용하여 애플리케이션 복원"
- "고급 복원 설정 사용"
- "NetApp SnapMirror 및 Trident Protect를 사용하여 애플리케이션 복제"
- "Trident Protect CLI 플러그인을 설치하세요"

- "AppVault 객체를 사용하여 버킷을 관리합니다"

Trident Protect 실행 후크 관리

실행 후크는 관리되는 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 따라서 애플리케이션 정합성이 보장되는 스냅샷이 보장됩니다.

실행 후크 유형

Trident Protect는 실행 가능 시기에 따라 다음 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 사후 스냅샷
- 사전 백업
- 백업 후
- 사후 복원
- 장애 조치 후

실행 순서

데이터 보호 작업이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. 해당되는 모든 사용자 정의 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 만들고 실행할 수 있지만, 이 후크의 실행 순서는 보장되거나 구성할 수 없습니다.
2. 해당되는 경우 파일 시스템이 정지됩니다. "Trident Protect를 사용하여 파일 시스템 동결을 구성하는 방법에 대해 자세히 알아보세요.".
3. 데이터 보호 작업이 수행됩니다.
4. 해당되는 경우 동결된 파일 시스템은 동결 해제됩니다.
5. 해당되는 모든 사용자 지정 작업 후 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 만들고 실행할 수 있지만 작업 후 후크의 실행 순서는 보장되거나 구성할 수 없습니다.

같은 유형의 실행 후크를 여러 개 생성하는 경우(예: 사전 스냅샷) 해당 후크의 실행 순서는 보장되지 않습니다. 그러나 다른 유형의 후크를 실행하는 순서는 보장됩니다. 예를 들어, 다음과 같은 순서로 서로 다른 유형의 후크가 모두 있는 구성을 실행할 수 있습니다.

1. 사전 스냅샷 후크가 실행되었습니다
2. 사후 스냅샷 후크가 실행되었습니다
3. 예비 후크가 실행되었습니다
4. 백업 후 후크가 실행되었습니다

참고	위의 순서 예제는 기존 스냅샷을 사용하지 않는 백업을 실행하는 경우에만 적용됩니다.

참고 | 운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 사용하도록 설정한 후 결과 스냅샷과 백업을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고, 스냅샷 또는 백업을 복원한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.

참고 | 스냅샷 사전 실행 후크가 Kubernetes 리소스를 추가, 변경 또는 제거하는 경우 스냅샷 또는 백업과 이후의 복원 작업에 변경 사항이 포함됩니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있습니다.
- Trident Protect에서는 실행 후크가 사용하는 스크립트가 실행 가능한 셸 스크립트 형식으로 작성되어야 합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Trident Protect는 실행 후크 설정과 일치 기준을 사용하여 스냅샷, 백업 또는 복원 작업에 적용할 수 있는 후크를 결정합니다.

참고 | 실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다. 연결된 실행 후크와 함께 백업 또는 스냅샷 작업을 시작한 다음 취소하면 백업 또는 스냅샷 작업이 이미 시작된 경우에도 후크를 실행할 수 있습니다. 즉, 백업 후 실행 후크에 사용되는 논리는 백업이 완료된 것으로 가정할 수 없습니다.

실행 후크 필터

응용 프로그램의 실행 후크를 추가하거나 편집할 때 실행 후크에 필터를 추가하여 후크가 일치할 컨테이너를 관리할 수 있습니다. 필터는 모든 컨테이너에서 동일한 컨테이너 이미지를 사용하는 응용 프로그램에 유용하지만 각 이미지를 다른 용도(예: Elasticsearch)로 사용할 수 있습니다. 필터를 사용하면 실행 후크가 실행되는 시나리오를 만들 수 있습니다. 단, 모든 동일한 컨테이너를 실행하는 것은 아닙니다. 단일 실행 후크에 대해 여러 개의 필터를 만들면 논리적 필터 및 연산자와 결합됩니다. 실행 후크당 최대 10개의 활성 필터를 사용할 수 있습니다.

실행 후크에 추가하는 각 필터는 정규 표현식을 사용하여 클러스터의 컨테이너와 일치시킵니다. 후크가 컨테이너와 일치하면 후크는 해당 컨테이너에서 연관된 스크립트를 실행합니다. 필터에 대한 정규 표현식은 RE2(Regular Expression 2) 구문을 사용하는데, 이 구문은 일치 항목 목록에서 컨테이너를 제외하는 필터를 만드는 것을 지원하지 않습니다. Trident Protect가 실행 후크 필터의 정규 표현식에 대해 지원하는 구문에 대한 정보는 다음을 참조하세요. "정규식 2(RE2) 구문 지원" .

참고 | 복원 또는 클론 작업 후에 실행되는 실행 후크에 네임스페이스 필터를 추가하고 복원 또는 클론 소스와 대상이 서로 다른 네임스페이스에 있는 경우 네임스페이스 필터는 대상 네임스페이스에만 적용됩니다.

실행 후크 예

를 방문하여 "NetApp Verda GitHub 프로젝트" Apache Cassandra 및 Elasticsearch와 같은 인기 앱의 실제 실행 후크를 다운로드하십시오. 예제를 보고 사용자 지정 실행 후크를 구조화하는 아이디어를 얻을 수도 있습니다.

참고 | Verda 저장소의 스크립트는 있는 그대로 제공되며 NetApp에서 공식적으로 지원하지 않습니다.

> NetApp 지원은 Trident Protect 내의 실행 후크 프레임워크에만 해당됩니다.

실행 후크를 만듭니다

Trident Protect를 사용하여 앱에 대한 사용자 정의 실행 후크를 만들 수 있습니다. 실행 후크를 생성하려면 소유자, 관리자 또는 멤버 권한이 필요합니다.

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-hook.yaml`지정합니다.
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.applicationRef *:(*required*) 실행 후크를 실행할 응용 프로그램의 Kubernetes 이름입니다.
   - * spec.stage *: (*required*) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 가능한 값:
     - 사전
     - 게시
   - * spec.action *:(*required*) 지정된 실행 후크 필터가 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
     - 스냅샷
     - 백업
     - 복원
     - 페일오버
   - **spec.enabled**: (*Optional*) 이 실행 후크의 활성화 여부를 나타냅니다. 지정하지 않으면 기본값은 true 입니다.
   - **spec.hookSource**:(*required*) base64로 인코딩된 후크 스크립트를 포함하는 문자열입니다.
   - **spec.timeout**: (*Optional*) 실행 후크가 실행될 수 있는 시간을 분 단위로 정의하는 숫자입니다. 최소값은 1분이고, 지정하지 않은 경우 기본값은 25분입니다.
   - **spec.arguments**: (*Optional*) 실행 후크에 지정할 수 있는 인수의 YAML 목록입니다.
   - * spec.matchingCriteria *: (*Optional*) 각 쌍이 실행 후크 필터를 구성하는 조건 키 값 쌍의 선택적 목록입니다. 실행 후크당 최대 10개의 필터를 추가할 수 있습니다.
   - * spec.matchingCriteria.type *: (*Optional*) 실행 후크 필터 유형을 식별하는 문자열입니다. 가능한 값:
     - 컨테이너이미지
     - 컨테이너명
     - PodName을 선택합니다
     - PodLabel을 선택합니다
     - 이름 이름 이름

◦ * spec.matchingCriteria.value *: (*Optional*) 실행 후크 필터 값을 식별하는 문자열 또는 정규식입니다.

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 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-hook.yaml
```

CLI를 사용합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 실행 후크를 만듭니다. 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>
```

실행 후크를 수동으로 실행합니다

테스트 목적으로 실행 후크를 수동으로 실행하거나 실패 후 수동으로 후크를 다시 실행해야 하는 경우 실행할 수 있습니다. 실행 후크를 수동으로 실행하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

실행 후크를 수동으로 실행하는 작업은 다음 두 가지 기본 단계로 구성됩니다.

1. 리소스 백업을 생성하여 리소스를 수집하고 해당 백업을 생성하여 후크가 실행될 위치를 결정합니다
2. 백업에 대해 실행 후크를 실행합니다

▼ **1**단계**:** 리소스 백업을 생성합니다

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-resource-backup.yaml`지정합니다.
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.applicationRef *:(*required*) 리소스 백업을 생성할 애플리케이션의 Kubernetes 이름입니다.
   - * spec.appVaultRef *: (*required*) 백업 내용이 저장된 AppVault의 이름입니다.
   - * spec.appArchivePath * : 백업 컨텐츠가 저장되는 AppVault 내부의 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 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

CLI를 사용합니다

단계

1. 대괄호 안의 값을 사용자 환경의 정보로 대체하여 백업을 만듭니다. 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 봅니다. 작업이 완료될 때까지 이 예제 명령을 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>
```

▼ 단계 **2:** 실행 후크를 실행합니다

CR을 사용합니다

단계

1. CR(사용자 정의 리소스) 파일을 만들고 이름을 `trident-protect-hook-run.yaml`지정합니다.
2. Trident Protect 환경 및 클러스터 구성에 맞게 다음 속성을 구성하세요.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.applicationRef *: (*required*) 이 값이 1단계에서 만든 ResourceBackup CR의 응용 프로그램 이름과 일치하는지 확인합니다.
   - * spec.appVaultRef *: (*required*) 이 값이 1단계에서 작성한 ResourceBackup CR의 appVaultRef와 일치하는지 확인합니다.
   - **spec.appArchivePath**: 이 값이 1단계에서 만든 ResourceBackup CR의 appArchivePath 와 일치하는지 확인하십시오.

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

- * spec.action *:(*required*) 지정된 실행 후크 필터가 일치한다고 가정할 때 실행 후크가 수행할 작업을 나타내는 문자열입니다. 가능한 값:
  - 스냅샷
  - 백업
  - 복원
  - 페일오버
- * spec.stage *: (*required*) 실행 후크가 실행되어야 하는 작업 중 단계를 나타내는 문자열입니다. 이 후크 실행은 다른 단계에서 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 파일을 올바른 값으로 채운 후 CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

CLI를 사용합니다

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. execution hook run의 상태를 확인한다. 작업이 완료될 때까지 이 명령을 반복적으로 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 Protect 제거

평가판에서 정식 버전으로 업그레이드하는 경우 Trident Protect 구성 요소를 제거해야 할 수도 있습니다.

Trident Protect를 제거하려면 다음 단계를 수행하세요.

단계

1. Trident Protect CR 파일을 제거합니다.

참고	이 단계는 25.06 이상 버전에서는 필요하지 않습니다.

```
helm uninstall -n trident-protect trident-protect-crds
```

2. Trident Protect 제거:

```
helm uninstall -n trident-protect trident-protect
```

3. Trident Protect 네임스페이스를 제거합니다.

```
kubectl delete ns trident-protect
```

Trident 및 Trident 프로텍트 블로그

훌륭한 NetApp Trident 및 Trident Protect 블로그는 다음에서 찾을 수 있습니다.

Trident 블로그

- 2025년 10월 16일:"Kubernetes를 위한 고급 스토리지 솔루션"
- 2025년 8월 19일:"데이터 일관성 향상: Trident 사용한 OpenShift 가상화의 볼륨 그룹 스냅샷"
- 2025년 5월 9일: "Amazon EKS 애드온을 사용한 FSx for ONTAP에 대한 자동 Trident 백엔드 구성"
- 2025년 4월 15일: "Google Cloud NetApp Volumes for SMB Protocol을 갖춘 NetApp Trident"
- 2025년 4월 14일: "Kubernetes에서 영구 저장소를 위해 Trident 25.02와 함께 파이버 채널 프로토콜 활용"
- 2025년 4월 14일: "Kubernetes 블록 스토리지를 위한 NetApp ASA r2 시스템의 잠재력 활용"
- 2025년 3월 31일: "새로운 인증 운영자를 통해 Red Hat OpenShift에서 Trident 설치 간소화"
- 2025년 3월 27일: "Google Cloud NetApp Volumes 사용하여 SMB용 Trident 프로비저닝"
- 2025년 3월 5일: "원활한 iSCSI 스토리지 통합의 실현: AWS용 Rosa Cluster의 FSxN에 대한 가이드"
- 2025년 2월 27일: "Trident, GKE 및 Google Cloud NetApp Volumes 사용하여 클라우드 ID 배포"
- 2024년 12월 12일: "Trident에서 파이버 채널 지원을 소개합니다"
- 2024년 11월 11일: "Google Cloud NetApp Volumes 탑재한 NetApp Trident"
- 2024년 10월 29일: "Trident 사용하여 AWS(ROSA)에서 Red Hat OpenShift 서비스를 사용하는 Amazon FSx for NetApp ONTAP"
- 2024년 10월 29일: "ROSA 및 Amazon FSx for NetApp ONTAP 에서 OpenShift Virtualization을 사용한 VM 라이브 마이그레이션"
- 2024년 7월 8일: "Amazon EKS에서 최신 컨테이너화된 앱에 대한 ONTAP 스토리지를 사용하기 위해 NVMe/TCP 사용"
- 2024년 7월 1일: "Google Cloud NetApp Volumes Flex 및 Astra Trident 활용한 원활한 Kubernetes 스토리지"
- 2024년 6월 11일: "OpenShift의 통합 이미지 레지스트리를 위한 백엔드 스토리지로서의 ONTAP"

Trident 프로텍트 블로그

- 2025년 5월 16일:"Trident Protect 복원 후 후크를 사용하여 재해 복구를 위한 레지스트리 장애 조치 자동화"
- 2025년 5월 16일: "NetApp Trident Protect를 통한 OpenShift 가상화 재해 복구"
- 2025년 5월 13일:"Trident Protect 백업 및 복원을 통한 스토리지 클래스 마이그레이션"
- 2025년 5월 9일:"Trident Protect 복원 후 후크를 사용하여 Kubernetes 애플리케이션 크기 조정"
- 2025년 4월 3일: "Trident Protect Power Up: 보호 및 재해 복구를 위한 Kubernetes 복제"
- 2025년 3월 13일: "OpenShift 가상화 VM을 위한 충돌 시에도 정합성이 보장되는 백업 및 복원 작업"
- 2025년 3월 11일: "NetApp Trident를 통해 GitOps 패턴을 애플리케이션 데이터 보호로 확장"
- 2025년 3월 3일: "Trident 25.02: Red Hat OpenShift 환경을 개선하는 새로운 기능"

- 2025년 1월 15일:"Trident Protect 역할 기반 액세스 제어 소개"
- 2024년 11월 11일:"Kubernetes 기반 데이터 관리: Trident Protect로 시작하는 새로운 시대"

지식 및 지원

자주 묻는 질문

Trident 설치, 구성, 업그레이드 및 문제 해결에 대한 질문과 대답을 찾아보십시오.

일반적인 질문

Trident를 얼마나 자주 릴리즈합니까**?**

24.02 릴리즈부터 Trident는 2월, 6월, 10월 등 4개월마다 릴리즈됩니다.

Trident는 특정 버전의 **Kubernetes**에서 릴리즈된 모든 기능을 지원합니까**?**

Trident은 일반적으로 Kubernetes에서 알파 기능을 지원하지 않습니다. Trident는 Kubernetes 베타 릴리즈를 따르는 두 Trident 릴리스 내에서 베타 기능을 지원할 수 있습니다.

Trident가 작동하는 데 다른 **NetApp** 제품에 종속성이 있습니까**?**

Trident는 다른 NetApp 소프트웨어 제품에 의존하지 않으며 독립 실행형 애플리케이션으로 작동합니다. 그러나 NetApp 백엔드 스토리지 디바이스가 있어야 합니다.

Trident 구성에 대한 전체 세부 정보를 어떻게 얻을 수 있습니까**?**

명령을 사용하여 `tridentctl get` Trident 구성에 대한 자세한 정보를 얻을 수 있습니다.

Trident를 통해 스토리지를 프로비저닝하는 방법에 관한 메트릭을 확인할 수 있습니까**?**

예. Prometheus 끝점은 관리되는 백 엔드 수, 프로비저닝된 볼륨 수, 사용된 바이트 등 Trident 작업에 대한 정보를 수집하는 데 사용할 수 있습니다. 모니터링 및 분석에도 를 "Cloud Insights"사용할 수 있습니다.

Trident를 **CSI Provisioner**로 사용할 때 사용자 환경이 달라집니까**?**

아니요. 사용자 환경 및 기능에 관한 한 변경 사항은 없습니다. 사용된 Provisioner 이름은 입니다 `csi.trident.netapp.io`. 이 Trident 설치 방법은 현재 및 향후 릴리스에서 제공하는 모든 새 기능을 사용하려는 경우에 권장됩니다.

Kubernetes 클러스터에 Trident를 설치하고 사용합니다

Trident는 개인 레지스트리에서 오프라인 설치를 지원합니까**?**

예, Trident는 오프라인으로 설치할 수 있습니다. 다음을 참조하십시오. "Trident 설치에 대해 알아봅니다".

Trident를 원격으로 설치할 수 있습니까**?**

예. Trident 18.10 이상에서는 클러스터에 액세스할 수 있는 모든 컴퓨터에서 원격 설치 기능을 `kubectl` 지원합니다. 액세스가 확인된 후 `kubectl`(예: 원격 컴퓨터에서 명령을 시작하여 `kubectl get nodes` 확인) 설치 지침을 따릅니다.

Trident로 고가용성을 구성할 수 있습니까**?**

Trident는 하나의 인스턴스가 있는 Kubernetes 배포(ReplicaSet)로 설치되므로 HA가 내장되어 있습니다. 배포에서 복제 수를 늘리면 안 됩니다. Trident가 설치된 노드가 손실되거나 Pod에 액세스할 수 없는 경우 Kubernetes는 이 Pod를 클러스터의 정상 노드로 자동으로 재구축합니다. Trident는 컨트롤 플레인만 해당하므로, 현재 마운트된 Pod를 Trident 재배포하면 영향을 받지 않습니다.

Trident는 **kube** 시스템 네임스페이스에 액세스해야 합니까**?**

Trident는 애플리케이션이 새로운 PVC를 요청하는 시기를 결정하기 위해 Kubernetes API 서버에서 읽으므로, kube-시스템에 액세스해야 합니다.

Trident에서 사용하는 역할 및 **Privileges**는 무엇입니까**?**

Trident 설치 프로그램은 Kubernetes ClusterRole을 생성하며, 이 역할은 클러스터의 PersistentVolume, PersistentVolumeClaim, StorageClass 및 Secret 리소스에 대한 특정 액세스 권한을 가집니다. "tridentctl 설치를 사용자 지정합니다"을 참조하십시오.

Trident에서 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성할 수 있습니까**?**

필요한 경우 Trident가 설치에 사용하는 정확한 매니페스트 파일을 로컬에서 생성하고 수정할 수 있습니다. "tridentctl 설치를 사용자 지정합니다"을 참조하십시오.

2개의 개별 **Kubernetes** 클러스터에 대해 **2**개의 개별 **Trident** 인스턴스에 대해 동일한 **ONTAP** 백엔드 **SVM**을 공유할 수 있습니까**?**

권장하지는 않지만 두 개의 Trident 인스턴스에 대해 동일한 백엔드 SVM을 사용할 수 있습니다. 설치하는 동안 각 인스턴스에 대해 고유한 볼륨 이름을 지정하거나 파일에 고유한 `StoragePrefix` 매개 변수를 `setup/backend.json` 지정합니다. 이는 두 인스턴스에 대해 동일한 FlexVol volume를 사용하지 않도록 하기 위한 것입니다.

ContainerLinux(이전의 **CoreOS)**에서 **Trident**를 설치할 수 있습니까**?**

Trident는 단순한 Kubernetes Pod로, Kubernetes가 실행 중인 모든 곳에 설치할 수 있습니다.

Trident를 **NetApp Cloud Volumes ONTAP**와 함께 사용할 수 있습니까**?**

예, Trident는 AWS, Google Cloud, Azure에서 지원됩니다.

문제 해결 및 지원

NetApp는 **Trident**를 지원합니까**?**

Trident는 오픈 소스이며 무료로 제공되지만, NetApp 백엔드가 지원되는 경우 NetApp는 이를 완벽하게 지원합니다.

지원 케이스를 어떻게 제기합니까**?**

지원 케이스를 제기하려면 다음 중 하나를 수행합니다.

1. 지원 어카운트 매니저에게 연락하여 티켓을 발급하는 데 도움을 받으십시오.
2. 에 연락하여 지원 케이스를 제출하십시오 "NetApp 지원".

지원 로그 번들을 생성하려면 어떻게 해야 합니까?

tridentctl logs-a를 실행하여 지원 번들을 생성할 수 있습니다. 번들에 캡처된 로그 외에 kubelet 로그를 캡처하여 Kubernetes 측의 마운트 문제를 진단합니다. 쿠벨렛 로그를 얻는 지침은 Kubernetes 설치 방법에 따라 다릅니다.

새 기능에 대한 요청을 제기해야 하는 경우 어떻게 해야 합니까?

문제를 "GitTrident 허브" 작성하고 문제에 대한 주제 및 설명에 * RFE * 를 언급합니다.

결함은 어디에서 제기합니까?

에서 문제를 "GitTrident 허브" 생성합니다. 문제와 관련된 모든 필수 정보와 로그를 포함해야 합니다.

Trident에 대한 빠른 질문이 있는 경우 어떻게 됩니까? 커뮤니티나 포럼이 있습니까?

질문, 문제 또는 요청이 있는 경우 Trident 또는 GitHub를 통해 "불화 채널"문의하십시오.

스토리지 시스템의 암호가 변경되어 **Trident**가 더 이상 작동하지 않습니다. 어떻게 복구합니까?

백엔드의 암호를 로 업데이트합니다 `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. 대치 `myBackend` 백엔드 이름을 포함하는 예에서 및 입니다 `/path/to_new_backend.json` 올바른 경로를 사용하여 `backend.json` 파일.

Trident이 내 **Kubernetes** 노드를 찾을 수 없습니다. 이 문제를 해결하려면 어떻게 합니까?

Trident이 Kubernetes 노드를 찾을 수 없는 두 가지 가능한 시나리오가 있습니다. Kubernetes의 네트워킹 문제 또는 DNS 문제 때문일 수 있습니다. 각 Kubernetes 노드에서 실행되는 Trident 노드 데모는 Trident 컨트롤러와 통신하여 노드를 Trident에 등록할 수 있어야 합니다. Trident를 설치한 후 네트워킹이 변경되면 클러스터에 추가되는 새로운 Kubernetes 노드에서만 이 문제가 발생합니다.

Trident POD가 제거되면 데이터를 손실합니까?

Trident POD를 제거할 경우 데이터가 손실되지 않습니다. Trident 메타데이터는 CRD 개체에 저장됩니다. Trident에서 프로비저닝한 모든 PVS가 정상적으로 작동합니다.

Trident를 업그레이드합니다

이전 버전에서 새 버전으로 직접 업그레이드할 수 있습니까(일부 버전을 건너뛰는 경우)?

NetApp는 하나의 주요 릴리즈에서 다음 주요 릴리즈로 Trident 업그레이드를 지원합니다. 버전 18.xx에서 19.xx, 19.xx에서 20.xx로 업그레이드할 수 있습니다. 운영 구축 전에 연구소에서 업그레이드를 테스트해야 합니다.

Trident를 이전 릴리즈로 다운그레이드할 수 있습니까?

업그레이드, 종속성 문제 또는 실패하거나 불완전한 업그레이드 후에 발견된 버그에 대한 수정이 필요한 경우 "Trident를 제거합니다"해당 버전에 대한 특정 지침을 사용하여 이전 버전을 다시 설치해야 합니다. 이 방법은 이전 버전으로 다운그레이드하는 유일한 권장 방법입니다.

백엔드 및 볼륨 관리

ONTAP 백엔드 정의 파일에서 관리와 **DataLIF**를 모두 정의해야 합니까**?**

관리 LIF는 필수입니다. DataLIF는 다양합니다.

- ONTAP SAN: iSCSI에 대해 지정하지 마십시오. Trident는 를 사용하여 "ONTAP 선택적 LUN 맵"다중 경로 세션을 설정하는 데 필요한 iSCI LIF를 검색합니다. 이 명시적으로 정의된 경우 경고가 `dataLIF` 생성됩니다. 자세한 내용은 을 "ONTAP SAN 구성 옵션 및 예" 참조하십시오.
- ONTAP NAS:NetApp는 를 지정할 것을 `dataLIF` 권장합니다. 제공되지 않는 경우 Trident는 SVM에서 데이터 LIF를 가져옵니다. NFS 마운트 작업에 사용할 FQDN(정규화된 도메인 이름)을 지정할 수 있습니다. 이렇게 하면 라운드 로빈 DNS를 생성하여 여러 데이터 LIF의 로드 밸런싱을 수행할 수 있습니다. 자세한 내용은 을 참조하십시오"ONTAP NAS 구성 옵션 및 예"

Trident에서 **ONTAP** 백엔드에 대해 **CHAP**를 구성할 수 있습니까**?**

예. Trident는 ONTAP 백엔드에 대해 양방향 CHAP를 지원합니다. 이를 위해서는 백엔드 구성에 설정이 `useCHAP=true` 필요합니다.

Trident에서 엑스포트 정책을 관리하려면 어떻게 해야 합니까**?**

Trident는 20.04 버전부터 내보내기 정책을 동적으로 생성하고 관리할 수 있습니다. 따라서 스토리지 관리자는 백엔드 구성에서 하나 이상의 CIDR 블록을 제공할 수 있으며, 이러한 범위에 속하는 Trident 추가 노드 IP를 생성한 엑스포트 정책에 추가할 수 있습니다. 이러한 방식으로 Trident는 지정된 CIDR 내에서 IP가 있는 노드에 대한 규칙의 추가 및 삭제를 자동으로 관리합니다.

IPv6 주소를 **Management** 및 **DataLIF**에 사용할 수 있습니까**?**

Trident는 다음에 대한 IPv6 주소 정의를 지원합니다.

- `managementLIF` 및 `dataLIF` ONTAP NAS 백엔드의 경우
- `managementLIF` ONTAP SAN 백엔드의 경우 지정할 수 없습니다 `dataLIF` ONTAP SAN 백엔드에서

Trident가 IPv6에서 작동하려면 flag(설치용), `IPv6` (Trident 운영자용) 또는 `tridentTPv6` (Helm 설치용 `tridentctl`)를 사용하여 설치해야 `--use-ipv6` 합니다.

백엔드에서 관리 **LIF**를 업데이트할 수 있습니까**?**

예. 'tridentctl update backend' 명령을 사용하여 백엔드 관리 LIF를 업데이트할 수 있습니다.

백엔드에서 **DataLIF**를 업데이트할 수 있습니까**?**

DataLIF는 및 `ontap-nas-economy` 에서만 업데이트할 수 `ontap-nas` 있습니다.

Kubernetes용 **Trident**에서 여러 개의 백엔드를 생성할 수 있습니까**?**

Trident는 동일한 드라이버나 다른 드라이버를 사용하여 여러 개의 백엔드를 동시에 지원할 수 있습니다.

Trident는 백엔드 자격 증명을 어떻게 저장합니까**?**

Trident는 백엔드 자격 증명을 Kubernetes Secrets로 저장합니다.

Trident는 특정 백엔드를 어떻게 선택합니까**?**

백엔드 속성을 사용하여 클래스에 맞는 풀을 자동으로 선택할 수 없는 경우 특정 풀 세트를 선택하는 데 'toragePools' 및 'additionalStoragePools' 매개 변수가 사용됩니다.

Trident가 특정 백엔드에서 프로비저닝되지 않도록 하려면 어떻게 해야 합니까**?**

`excludeStoragePools`매개 변수는 Trident가 프로비저닝에 사용하는 풀 세트를 필터링하고 일치하는 풀을 모두 제거하는 데 사용됩니다.

동일한 종류의 백엔드가 여러 개 있는 경우 **Trident**는 사용할 백엔드를 어떻게 선택합니까**?**

동일한 유형의 백엔드가 여러 개 구성된 경우 Trident는 및 `PersistentVolumeClaim` 에 있는 매개 변수를 기반으로 적절한 백엔드를 선택합니다 `StorageClass`. 예를 들어 ONTAP-NAS 드라이버 백엔드가 여러 개 있는 경우 Trident는 및 `PersistentVolumeClaim` 의 매개 변수를 일치시키고 백엔드를 일치시키려고 시도하며 `StorageClass`, 이 백엔드는 및 `PersistentVolumeClaim` 에 나열된 요구 사항을 제공할 수 있습니다 `StorageClass`. 요청과 일치하는 백엔드가 여러 개 있는 경우 Trident는 해당 백엔드에서 임의의 백엔드를 선택합니다.

Trident는 요소**/SolidFire**가 있는 양방향 **CHAP**를 지원합니까**?**

예.

Trident에서는 **ONTAP** 볼륨에서 **qtree**를 어떻게 배포합니까**?** 단일 볼륨에 몇 개의 **qtree**를 구축할 수 있습니까**?**

이 `ontap-nas-economy` 드라이버는 동일한 FlexVol volume에 최대 200개의 qtree(50~300개 범위에서 구성 가능), 클러스터 노드당 100,000개 qtree, 클러스터당 2.4M을 생성합니다. 이코노미 드라이버에서 서비스되는 새 항목을 입력하면 `PersistentVolumeClaim` 운전자는 새 Qtree를 서비스할 수 있는 FlexVol volume가 이미 있는지 확인합니다. Qtree를 서비스할 수 있는 FlexVol volume가 존재하지 않으면 새 FlexVol volume가 생성됩니다.

ONTAP NAS에 프로비저닝된 볼륨에 대해 **Unix** 권한을 설정하려면 어떻게 해야 합니까**?**

백엔드 정의 파일에 매개 변수를 설정하여 Trident에서 프로비저닝한 볼륨에 대해 Unix 권한을 설정할 수 있습니다.

볼륨을 프로비저닝하는 동안 명시적 **ONTAP NFS** 마운트 옵션 세트를 구성하려면 어떻게 합니까**?**

기본적으로 Trident은 Kubernetes에서 마운트 옵션을 아무 값으로 설정하지 않습니다. Kubernetes 스토리지 클래스에서 마운트 옵션을 지정하려면 지정된 예제를 "여기"따릅니다.

프로비저닝된 볼륨을 특정 엑스포트 정책으로 설정하려면 어떻게 해야 합니까**?**

적절한 호스트가 볼륨에 액세스할 수 있도록 하려면 백엔드 정의 파일에 구성된 "exportPolicy" 매개 변수를 사용합니다.

ONTAP 및 **Trident**를 통해 볼륨 암호화를 설정하려면 어떻게 해야 합니까**?**

백엔드 정의 파일의 암호화 매개 변수를 사용하여 Trident에서 프로비저닝한 볼륨에 대한 암호화를 설정할 수 있습니다. 자세한 내용은 다음을 참조하십시오. "Trident가 NVE 및 NAE와 작동하는 방법"

Trident를 통해 **ONTAP**용 **QoS**를 구현하는 가장 좋은 방법은 무엇입니까**?**

ONTAP용 QoS를 구현하려면 'torageClaes'를 사용합니다.

Trident를 통해 씬 또는 일반 프로비저닝을 지정하려면 어떻게 해야 합니까**?**

ONTAP 드라이버는 씬 또는 일반 프로비저닝을 지원합니다. ONTAP 드라이버는 기본적으로 씬 프로비저닝입니다. 일반 프로비저닝이 필요한 경우 백엔드 정의 파일 또는 'torageClass'를 구성해야 합니다. 둘 다 구성된 경우 'torageClass'가 우선합니다. ONTAP에 대해 다음을 구성합니다.

1. 'torageClass'에서 'vorioningType' 속성을 굵게로 설정합니다.
2. 백엔드 정의 파일에서 'Backend spaceReserve Parameter'를 볼륨으로 설정하여 일반 볼륨을 활성화합니다.

실수로 **PVC**를 삭제한 경우에도 사용 중인 볼륨이 삭제되지 않도록 하려면 어떻게 해야 합니까**?**

PVC 보호는 버전 1.10부터 Kubernetes에서 자동으로 활성화됩니다.

Trident에서 생성된 **NFS PVC**를 확장할 수 있습니까**?**

예. Trident에서 생성한 NFS PVC를 확장할 수 있습니다. Trident 자동 확장 정책은 용량 임계값에 따라 PVC를 자동으로 확장할 수 있습니다. ONTAP 볼륨 자동 크기 조정은 별개이며 지원되는 ONTAP SAN 드라이버에 적용됩니다. 이는 PVC 크기를 변경하지 않고 백업 FlexVol을 확장합니다. 자세한 내용은 "ONTAP 볼륨 자동 크기 조정 "을 참조하십시오.

SnapMirror 데이터 보호**(DP)** 또는 오프라인 모드일 때 볼륨을 가져올 수 있습니까**?**

외부 볼륨이 DP 모드이거나 오프라인인 경우 볼륨 가져오기가 실패합니다. 다음과 같은 오류 메시지가 나타납니다.

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

리소스 할당량은 **NetApp** 클러스터로 어떻게 변환됩니까**?**

NetApp 스토리지의 용량이 있는 경우 Kubernetes 스토리지 리소스 할당량이 작동합니다. 용량 부족으로 인해 NetApp 스토리지가 Kubernetes 할당량 설정을 적용할 수 없는 경우, Trident은 프로비저닝을 시도하지만 오류를 생성합니다.

Trident를 사용하여 볼륨 스냅샷을 생성할 수 있습니까**?**

예. Trident에서는 주문형 볼륨 스냅샷 및 스냅샷에서 영구 볼륨 생성이 지원됩니다. 스냅샷에서 PVS를 생성하려면 `VolumeSnapshotDataSource` Feature Gate가 활성화되어 있는지 확인합니다.

Trident 볼륨 스냅샷을 지원하는 드라이버는 무엇입니까**?**

오늘부터 주문형 스냅샷 지원이 가능합니다. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , 그리고 `azure-netapp-files` 백엔드 드라이버.

ONTAP을 통해 Trident에서 프로비저닝한 볼륨의 스냅샷 백업을 어떻게 생성합니까?

ONTAP-NAS, ONTAP-SAN, ONTAP-NAS-Flexgroup 드라이버에서 지원됩니다. FlexVol 레벨에서 ONTAP-SAN-이코노미 드라이버에 대한 스냅샷 정책을 지정할 수도 있습니다.

드라이버에서는 사용할 수도 있지만 FlexVol volume 레벨의 세분화된 레벨에서는 사용할 수 `ontap-nas-economy` 없습니다. Trident에서 프로비저닝한 볼륨에 대해 스냅샷을 생성할 수 있도록 하려면 backend 매개 변수 옵션을 `snapshotPolicy` ONTAP 백엔드에 정의된 대로 원하는 스냅샷 정책으로 설정합니다. 스토리지 컨트롤러에서 생성한 모든 스냅샷은 Trident에서 알 수 없습니다.

Trident를 통해 프로비저닝된 볼륨에 대해 스냅샷 예약 비율을 설정할 수 있습니까?

예. 백엔드 정의 파일에 속성을 설정하여 Trident를 통해 스냅샷 복사본을 저장하기 위해 특정 비율의 디스크 공간을 예약할 수 `snapshotReserve` 있습니다. 백엔드 정의 파일에 및 을 `snapshotReserve` 구성한 경우 `snapshotPolicy` 스냅숏 예비 공간 백분율은 백엔드 파일에 언급된 백분율에 따라 `snapshotReserve` 설정됩니다. 백분율 숫자가 언급되지 않은 경우 `snapshotReserve` ONTAP는 기본적으로 스냅숏 예비 공간 비율을 5로 설정합니다. 이 `snapshotPolicy` 옵션을 none으로 설정하면 스냅숏 예비 공간 비율이 0으로 설정됩니다.

볼륨 스냅샷 디렉토리에 직접 액세스하고 파일을 복사할 수 있습니까?

예. 백엔드 정의 파일에서 '스냅샷 디렉토리' 매개 변수를 설정하여 Trident에서 프로비저닝한 볼륨의 스냅샷 디렉토리에 액세스할 수 있습니다.

Trident를 통해 볼륨에 SnapMirror를 설정할 수 있습니까?

현재 ONTAP CLI 또는 OnCommand System Manager를 사용하여 외부에서 SnapMirror를 설정해야 합니다.

영구 볼륨을 특정 ONTAP 스냅샷으로 복원하려면 어떻게 합니까?

ONTAP 스냅숏에 볼륨을 복원하려면 다음 단계를 수행하십시오.

1. 영구 볼륨을 사용하는 응용 프로그램 포드를 중지합니다.
2. ONTAP CLI 또는 OnCommand System Manager를 통해 필요한 스냅샷으로 되돌립니다.
3. 응용 프로그램 포드를 다시 시작합니다.

Trident가 로드 공유 미러가 구성된 SVM에서 볼륨을 프로비저닝할 수 있습니까?

NFS를 통해 데이터를 제공하는 SVM의 루트 볼륨에 로드 공유 미러를 생성할 수 있습니다. ONTAP는 Trident에서 생성한 볼륨의 로드 공유 미러를 자동으로 업데이트합니다. 이로 인해 볼륨 마운팅이 지연될 수 있습니다. Trident를 사용하여 여러 볼륨을 생성할 경우 볼륨 프로비저닝은 ONTAP에서 로드 공유 미러 업데이트에 따라 달라집니다.

각 고객/테넌트에 대해 스토리지 클래스 사용을 어떻게 분리할 수 있습니까?

Kubernetes에서는 네임스페이스의 스토리지 클래스를 허용하지 않습니다. 그러나 Kubernetes를 사용하여 네임스페이스당 사용되는 스토리지 리소스 할당량을 사용하여 네임스페이스당 특정 스토리지 클래스의 사용을 제한할 수 있습니다. 특정 스토리지에 대한 특정 네임스페이스 액세스를 거부하려면 해당 스토리지 클래스에 대한 리소스 할당량을 0으로 설정합니다.

문제 해결

Trident를 설치하고 사용하는 동안 발생할 수 있는 문제를 해결하려면 여기에 제공된 포인터를 사용하십시오.

참고	Trident에 대한 도움이 필요하면 를 사용하여 지원 번들을 만든 `tridentctl logs -a -n trident` 후 NetApp 지원 팀에 보내십시오.

일반 문제 해결

- Trident 포드가 제대로 표시되지 않으면(예: 두 개 미만의 준비된 컨테이너로 'ContainerCreating' 단계에 Trident 포드가 들러붙은 경우) kubeck-n trident t덴트 설명 배포 트리덴트를 실행하고 'kebctl-n trident tor stand' kudl- 를 실행합니다 이 밖에도 많은 정보를 얻을 수 있습니다. 쿠벨렛 로그(예: 저널링ctl-xeu kubelet)를 얻는 것도 도움이 될 수 있습니다.
- Trident 로그에 충분한 정보가 없으면 설치 옵션에 따라 설치 매개 변수에 '-d' 플래그를 전달하여 Trident에 대한 디버그 모드를 활성화할 수 있습니다.

  그런 다음 디버그가 "./tridentctl logs -n triment"를 사용하여 설정되었는지 확인하고 로그에서 level=debug msg를 검색합니다.

  운영자와 함께 설치됩니다

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  그러면 모든 Trident 포드가 다시 시작됩니다. 이 작업은 몇 초 정도 걸릴 수 있습니다. 'kubeck get pod-n trident'의 출력에서 '나이' 열을 관찰하여 이를 확인할 수 있습니다.

  Trident 20.07 및 20.10의 경우 `tprov` 대신 `torc` 을 사용하십시오.

  헬름과 함께 설치

  ```
  helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  tridentctl과 함께 설치됩니다

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- 백엔드 정의에 포함하여 각 백엔드에 대한 디버그 로그를 얻을 수도 `debugTraceFlags` 있습니다. 예를 들어, Trident 로그에 API 호출 및 메서드 트래버스를 가져오려면 를 `debugTraceFlags: {"api":true, "method":true,}` 포함합니다. 기존 백엔드는 로 `tridentctl backend update`구성할 수 `debugTraceFlags` 있습니다.
- RHCOS(Red Hat Enterprise Linux CoreOS)를 사용하는 경우 작업자 노드에서 이 `iscsid` 활성화되고 기본적으로 시작되는지 확인합니다. 이 작업은 OpenShift MachineConfigs를 사용하거나 점화 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 rpcbind가 설치되어 실행되고 있는지 확인합니다. 호스트 OS에 필요한 패키지 관리자를 사용하고 rpcbind가 실행 중인지 확인합니다. 'stemctl status rpcbind' 또는 이와 동등한 기능을 실행하여 rpcbind 서비스의 상태를 확인할 수 있습니다.
- Trident 백엔드가 이전에 작업을 수행했음에도 불구하고 "실패" 상태에 있다고 보고할 경우 백엔드와 연결된 SVM/관리 자격 증명을 변경하면 원인일 수 있습니다. 'tridentctl update backend'를 사용하여 백엔드 정보를 업데이트하거나 Trident POD를 바운딩하면 이 문제가 해결됩니다.
- Docker를 컨테이너 런타임으로 사용하여 Trident를 설치할 때 권한 문제가 발생하면 '--in cluster=false' 플래그를 사용하여 Trident를 설치해 보십시오. 설치자 포드는 사용하지 않고 설치자 이용으로 인한 권한 문제를 피한다.
- 실패한 실행 후 정리 작업을 위해 'uninstall parameter <uninstall Trident>'를 사용합니다. 기본적으로 이 스크립트는 Trident에서 만든 CRD를 제거하지 않으므로 실행 중인 구축에서도 안전하게 제거한 후 다시 설치할 수 있습니다.
- Trident의 이전 버전으로 다운그레이드하려면 먼저 를 실행하십시오 `tridentctl uninstall` Trident를 제거하는 명령입니다. 원하는 를 다운로드합니다 "Trident 버전" 를 사용하여 를 설치합니다 `tridentctl install` 명령.
- 설치가 성공적으로 완료된 후 PVC가 보류 단계에 고착되면 kubeck tl t설명해 PVC를 실행하면 Trident가 이 PVC에 대한 PV를 프로비저닝하지 못한 이유에 대한 추가 정보가 제공됩니다.

연산자를 사용한 **Trident** 배포 실패

연산자를 사용하여 Trident를 배포하는 경우 트리펜터터의 상태가 Installing에서 Installed로 변경됩니다. 'Failed(실패)' 상태를 확인하고 운용자가 자체적으로 복구할 수 없는 경우 다음 명령어를 실행해 운용자의 로그를 확인해야 한다.

```
tridentctl logs -l trident-operator
```

삼원 운영자 컨테이너의 로그를 뒤로하면 문제가 있는 위치를 가리킬 수 있습니다. 예를 들어, 이러한 문제 중 하나는 Airgapped 환경의 업스트림 등록부에서 필요한 컨테이너 이미지를 가져올 수 없는 것일 수 있습니다.

Trident의 설치가 실패한 이유를 이해하려면 '트리엔트오케스트레이터' 상태를 살펴보아야 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 '트리엔오케스트레이터'가 이미 있음을 나타냅니다. 각 Kubernetes 클러스터에는 Trident의 인스턴스가 하나만 있을 수 있으므로 운영자는 언제든지 생성할 수 있는 활성 'Trident Orchestrator'가 하나만 존재하도록 합니다.

또한 Trident Pod의 상태를 관찰하면 무언가 잘못되었음을 나타내는 경우가 많습니다.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

하나 이상의 컨테이너 이미지를 가져오지 않았기 때문에 포드를 완전히 초기화할 수 없다는 것을 분명히 알 수 있습니다.

이 문제를 해결하려면 트리엔오케스트레이터 CR을 편집해야 합니다. 또는 '트리엔오케스트레이터'를 삭제하고 수정되고 정확한 정의를 가진 새 정의를 만들 수 있습니다.

를 사용하여 Trident를 배포하지 못했습니다 `tridentctl`

무엇이 잘못되었는지 알 수 있도록 디버그 모드를 켜고 무엇이 문제인지 이해하는 데 도움이 되는 "-d' 인수를 사용하여 설치 프로그램을 다시 실행할 수 있습니다.

```
./tridentctl install -n trident -d
```

이 문제를 해결한 후 다음과 같이 설치를 정리한 다음 'tridentctl install' 명령을 다시 실행할 수 있습니다.

```
./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.

경고	이 작업은 취소할 수 없습니다. Trident를 완전히 새로 설치하려는 경우가 아니라면 이 작업을 수행하지 마십시오. CRD를 제거하지 않고 Trident를 제거하려면 을 "Trident를 제거합니다"참조하십시오.

Trident 운영자

Trident를 제거하고 Trident 운영자를 사용하여 CRD를 완전히 제거하려면:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

헬름

Helm을 사용하여 Trident를 제거하고 CRD를 완전히 제거하려면:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>tridentctl</code>

를 사용하여 Trident를 제거한 후 CRD를 완전히 제거합니다 `tridentctl`

```
tridentctl obliviate crd
```

rwx 원시 블록 네임스페이스와 관련된 NVMe 노드 스테이징 해제 실패 o Kubernetes 1.26

Kubernetes 1.26을 실행 중인 경우 rwx 원시 블록 네임스페이스와 함께 NVMe/TCP를 사용할 때 노드 스테이징 해제가 실패할 수 있습니다. 다음 시나리오는 오류에 대한 해결 방법을 제공합니다. 또는 Kubernetes를 1.27로 업그레이드할 수도 있습니다.

네임스페이스 및 Pod를 삭제했습니다

Pod에 Trident 관리 네임스페이스(NVMe 영구 볼륨)가 연결되어 있는 경우를 가정해 보겠습니다. ONTAP 백엔드에서 네임스페이스를 직접 삭제하는 경우, Pod를 삭제하려고 하면 스테이징 프로세스가 중단됩니다. 이 시나리오는 Kubernetes 클러스터나 다른 작동에 영향을 주지 않습니다.

해결 방법

해당 노드에서 영구 볼륨(해당 네임스페이스에 해당)을 마운트 해제하고 삭제합니다.

데이터 LIF가 차단되었습니다

```
 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
전체 기능을 복원하려면 dataLIFS를 불러옵니다.
```

네임스페이스 매핑을 삭제했습니다

```
 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
를 추가합니다 `hostNQN` 하위 시스템으로 돌아갑니다.
```

ONTAP 업그레이드 후 "v4.2-xattrs"가 활성화될 것으로 예상할 때 NFSv4.2 클라이언트가 "잘못된 인수"를 보고합니다.

ONTAP 업그레이드한 후 NFSv4.2 클라이언트가 NFSv4.2 내보내기를 마운트하려고 할 때 "잘못된 인수" 오류를 보고할 수 있습니다. 이 문제는 다음과 같은 경우에 발생합니다. `v4.2-xattrs` SVM에서는 옵션이 활성화되어 있지 않습니다. .해결 방법 활성화 `v4.2-xattrs` SVM에서 옵션을 선택하거나 ONTAP 9.12.1 이상으로 업그레이드하면 이 옵션이 기본적으로 활성화됩니다.

지원

NetApp은 다양한 방법으로 Trident를 지원합니다. 기술 자료(KB) 기사 및 불화 채널 같은 광범위한 무료 셀프 지원 옵션이 24x7 제공됩니다.

Trident 지원 라이프사이클

Trident는 버전에 따라 세 가지 수준의 지원을 제공합니다. 을 "정의를 위한 NetApp 소프트웨어 버전 지원" 참조하십시오.

완벽한 지원

Trident는 출시일로부터 12개월 동안 완벽한 지원을 제공합니다.

제한된 지원

Trident는 출시일로부터 13-24개월 동안 제한된 지원을 제공합니다.

자가 지원

Trident 문서는 출시일로부터 25-36개월 동안 사용할 수 있습니다.

버전	전면 지원 종료	제한적 지원 종료	자가 지원 종료
"26.10"	2027년 10월	2028년 10월	2029년 10월
"26.06"	2027년 7월	2028년 7월	2029년 7월
"26.02"	2027년 2월	2028년 2월	2029년 2월
"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
"2007년 3월"	—	—	2026년 7월
"23.04"	—	—	2026년 4월

자가 지원

문제 해결 문서의 전체 목록은 "NetApp Knowledgebase(로그인 필요)"을 참조하십시오.

커뮤니티 지원

당사의 컨테이너 사용자(Trident 개발자 포함)로 구성된 활발한 공용 커뮤니티가 "불화 채널"있습니다. 프로젝트에 대한 일반적인 질문을 하고 비슷한 생각을 가진 동료와 관련 주제를 논의할 수 있는 좋은 장소입니다.

NetApp 기술 지원

Trident에 대한 도움말을 보려면 를 사용하여 지원 번들을 만들어 `tridentctl logs -a -n trident` 로 `NetApp Support <Getting Help>`보내십시오.

를 참조하십시오

- "Trident 리소스"
- "쿠버네티스 허브"

참조하십시오

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.

참고 을 사용하여 설치하는 동안 활성/준비 프로브 포트를 변경할 수 있습니다 `--probe-port` 깃발. 이 포트가 작업자 노드의 다른 프로세스에서 사용되지 않도록 하는 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 API 사용 시기

REST API는 Kubernetes가 아닌 다른 구축 환경에서 Trident를 독립 실행형 바이너리로 사용하는 고급 설치에 유용합니다.

보안을 강화하기 위해 Trident는 `REST API` Pod 내부에서 실행될 때 기본적으로 localhost로 제한됩니다. 이 동작을 변경하려면 해당 POD 구성에서 Trident의 `-address` 인수를 설정해야 합니다.

REST API 사용

이러한 API가 호출되는 방법의 예를 보려면 DEBUG(`-d`) 플래그를 전달합니다. 자세한 내용은 을 "tridentctl을 사용하여 Trident를 관리합니다"참조하십시오.

API는 다음과 같이 작동합니다.

가져오기

`GET <trident-address>/trident/v1/<object-type>`

해당 형식의 모든 개체를 나열합니다.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

명명된 개체의 세부 정보를 가져옵니다.

게시

POST <trident-address>/trident/v1/<object-type>

지정된 형식의 개체를 만듭니다.

- 생성할 개체의 JSON 구성이 필요합니다. 각 개체 유형의 사양은 을 "tridentctl을 사용하여 Trident를 관리합니다"참조하십시오.
- 개체가 이미 있는 경우 동작이 달라집니다. backends는 기존 개체를 업데이트하지만 다른 모든 개체 형식은 작업에 실패합니다.

삭제

DELETE <trident-address>/trident/v1/<object-type>/<object-name>

명명된 리소스를 삭제합니다.

참고	백엔드 또는 스토리지 클래스와 연결된 볼륨은 계속 존재하므로 별도로 삭제해야 합니다. 자세한 내용은 을 "tridentctl을 사용하여 Trident를 관리합니다"참조하십시오.

명령줄 옵션

Trident는 Trident 오케스트레이터를 위한 몇 가지 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. 기본적으로 info(정보) 가 사용됩니다.

쿠버네티스

-k8s_pod

이 옵션 또는 을 사용합니다 `-k8s_api_server` Kubernetes 지원을 사용하도록 설정하십시오. 이 설정을 사용하면 Trident에서 포함된 POD의 Kubernetes 서비스 계정 자격 증명을 사용하여 API 서버에 연락합니다. Trident가 서비스 계정이 활성화된 Kubernetes 클러스터에서 POD로 실행되는 경우에만 작동합니다.

-k8s_api_server <insecure-address:insecure-port>

Kubernetes 지원을 활성화하려면 이 옵션 또는 `-k8s_pod` 을 사용합니다. Trident가 지정된 경우 제공된 비보안 주소 및 포트를 사용하여 Kubernetes API 서버에 연결합니다. 따라서 Trident는 Pod 외부에 배포할 수 있지만 API 서버에 대한 안전하지 않은 연결만 지원합니다. 안전하게 연결하려면 옵션을 사용하여 Trident를 Pod에 `-k8s_pod` 배포합니다.

Docker 를 참조하십시오

-volume_driver <name>

Docker 플러그인을 등록할 때 사용되는 드라이버 이름입니다. 기본값은 입니다 `netapp`.

-driver_port <port-number>

UNIX 도메인 소켓이 아닌 이 포트에서 수신 대기합니다.

-config <file>

필수 요소로서 백엔드 구성 파일에 대한 이 경로를 지정해야 합니다.

휴식

-address <ip-or-host>

Trident의 REST 서버가 수신해야 하는 주소를 지정합니다. 기본값은 localhost입니다. localhost에서 듣거나 Kubernetes Pod에서 실행 중인 경우, REST 인터페이스는 Pod 외부에서 직접 액세스할 수 없습니다. 사용 `-address ""` REST 인터페이스를 POD IP 주소에서 액세스할 수 있도록 합니다.

주의	Trident REST 인터페이스는 127.0.0.1(IPv4의 경우) 또는 [::1](IPv6의 경우)에서만 수신 및 서비스하도록 구성할 수 있습니다.

-port <port-number>

Trident의 REST 서버가 수신해야 하는 포트를 지정합니다. 기본값은 8000입니다.

-rest

REST 인터페이스를 활성화합니다. 기본값은 true 입니다.

Kubernetes 및 Trident 오브젝트

REST API를 사용하여 리소스 객체를 읽고 쓰면서 Kubernetes 및 Trident와 상호 작용할 수 있습니다. Kubernetes 및 Trident, Trident 및 스토리지와 Kubernetes 및 스토리지 간의 관계를 결정하는 몇 가지 리소스 개체가 있습니다. 이러한 오브젝트 중 일부는 Kubernetes를 통해 관리되며 나머지는 Trident를 통해 관리됩니다.

개체가 서로 어떻게 상호 작용합니까?

오브젝트, 목표 및 상호 작용 방식을 이해하는 가장 쉬운 방법은 Kubernetes 사용자의 스토리지에 대한 단일 요청을 따르는 것입니다.

1. 사용자가 이전에 관리자가 구성한 Kubernetes의 torageClass에서 특정 크기의 새 PersistentVolume을 요청하는 PersistentVolumeClaim을 만듭니다.
2. Kubernetes의 torageClass는 Trident를 프로비저닝자로 식별하고 Trident에 요청된 클래스에 대한 볼륨 프로비저닝 방법을 알려주는 매개 변수를 포함합니다.
3. Trident는 클래스에 대한 볼륨을 프로비저닝하는 데 사용할 수 있는 일치하는 'backends'와 'torагePools'를 식별하는 이름과 동일한 이름의 자체 'torageClass'를 찾습니다.
4. Trident는 일치하는 백엔드에 스토리지를 프로비저닝하고 Kubernetes의 "PersistentVolume"과 "PersistentVolume"과 실제 스토리지 간의 관계를 유지하는 Trident의 볼륨을 찾고, 마운트하고, 처리하는 방법을 Kubernetes의 "PersistentVolume"과 두 개의 객체를 만듭니다.

5. Kubernetes는 PersistentVolumeClaim을 새로운 PersistentVolume에 바인딩합니다. PersistentVolumeClaim이 실행되는 모든 호스트에서 PersistentVolume이 마운트되는 "PersistentVolumeClaim"이 포함된 POD
6. 사용자가 Trident를 가리키는 VolumeSnapshotClass를 사용하여 기존 PVC의 VolumeSnapshot을 생성합니다.
7. Trident는 PVC와 연결된 볼륨을 식별하고 백엔드에 볼륨의 스냅샷을 생성합니다. 또한 스냅샷을 식별하는 방법을 Kubernetes에 지시하는 'VolumeSnapshotContent'도 생성합니다.
8. 사용자는 'VolumeSnapshot'을 소스로 사용하여 'PersistentVolumeClaim'을 생성할 수 있습니다.
9. Trident는 필요한 스냅샷을 식별하고 "PersistentVolume"과 "Volume"을 생성하는 것과 동일한 단계를 수행합니다.

팁 Kubernetes 객체에 대한 자세한 내용은 를 읽는 것이 좋습니다 "영구 볼륨" 섹션을 참조하십시오.

쿠버네티스 `PersistentVolumeClaim` 오브젝트

Kubernetes "PersistentVolumeClaim" 개체는 Kubernetes 클러스터 사용자가 만든 스토리지 요청입니다.

Trident는 표준 사양 외에도 사용자가 백엔드 구성에서 설정한 기본값을 무효화하려는 경우 다음 볼륨별 주석을 지정할 수 있도록 합니다.

주석	볼륨 옵션	지원되는 드라이버
trident.netapp.io/fileSystem	파일 시스템	ONTAP-SAN, solidfire-SAN, ONTAP-SAN - 경제성
trident.netapp.io/cloneFromPVC	CloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	SplitOnClone 을 참조하십시오	ONTAP-NAS, ONTAP-SAN
trident.netapp.io/protocol	프로토콜	모두
trident.netapp.io/exportPolicy	내보내기 정책	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/snapshotPolicy	스냅샷 정책	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN
trident.netapp.io/snapshotReserve	snapshotReserve	온탑나스, 온탑나스플렉스그룹, 온탑산
trident.netapp.io/snapshotDirectory	스냅샷 디렉토리	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup
trident.netapp.io/blockSize	블록 크기	solidfire-SAN
트라이던트.넷앱.io/스킵리커버리큐	skipRecoveryQueue	온탑-나스, 온탑-나스-이코노미, 온탑-나스-플렉스그룹, 온탑-샌, 온탑-샌-이코노미

생성된 PV에 'Delete' Reclaim 정책이 있는 경우, Trident는 PV가 해제될 때(즉, 사용자가 PVC를 삭제할 때) PV와 보조 볼륨을 모두 삭제합니다. 삭제 작업이 실패할 경우 Trident는 PV를 해당 상태로 표시하고 성공할 때까지 또는 PV를 수동으로 삭제할 때까지 주기적으로 작업을 다시 시도합니다. PV에서 'Retain' 정책을 사용할 경우 Trident는 이를 무시하고 관리자가 Kubernetes 및 백엔드에서 이를 정리하여 제거하기 전에 볼륨을 백업 또는 검사할 수 있다고

가정합니다. PV를 삭제해도 Trident에서 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 PVS의 시점 복제본을 스냅샷 형태로 Kubernetes에 표시할 수 있습니다. 그런 다음 스냅샷을 사용하여 새 PVS를 생성할 수 있습니다. 이 기능이 어떻게 동작하는지 알아보려면 "주문형 볼륨 스냅샷"를 살펴보십시오.

Trident는 도 제공합니다 `cloneFromPVC` 및 `splitOnClone` 클론 생성을 위한 주석. CSI 구현을 사용하지 않고도 이러한 주석을 사용하여 PVC를 복제할 수 있습니다.

예를 들어 이미 mysql이라는 PVC가 있는 사용자는 trident.netapp.io/cloneFromPVC: mysql 같은 주석을 사용해 mysqlclone이라는 새로운 PVC를 만들 수 있습니다. 이 주석을 설정하면 Trident가 볼륨을 처음부터 프로비저닝하는 대신 MySQL PVC에 해당하는 볼륨을 클론합니다.

다음 사항을 고려하십시오.

- NetApp은 유휴 볼륨을 클론 복제할 것을 권장합니다.
- PVC와 그 클론은 동일한 Kubernetes 네임스페이스에서 동일한 스토리지 클래스를 가져야 합니다.
- ONTAP-NAS와 ONTAP-SAN 드라이버를 함께 사용하면 trident.netapp.io/splitOnClone` 과 함께 PVC 주석 trident.netapp.io/cloneFromPVC` 을 설정하는 것이 바람직할 수 있습니다. Trident는 trident.netapp.io/splitOnClone` 를 true로 설정하면 상위 볼륨에서 복제된 볼륨을 분할하여 복제된 볼륨의 수명 주기를 부모 볼륨에서 완전히 분리하여 스토리지 효율성을 잃게 됩니다. trident.netapp.io/splitOnClone` 를 설정하지 않거나 "false"로 설정하지 않으면 상위 볼륨과 클론 볼륨 간의 종속성을 생성하여 클론이 먼저 삭제되지 않으면 상위 볼륨을 삭제할 수 없게 되어 백엔드에서 공간 소비가 줄어듭니다. 클론을 분할하는 것이 올바른 시나리오는 빈 데이터베이스 볼륨을 복제하여 볼륨과 해당 클론이 크게 달라질 것으로 예상되며 ONTAP에서 제공하는 스토리지 효율성의 이점을 얻지 못하는 경우입니다.

를 클릭합니다 `sample-input` 디렉토리에는 Trident와 함께 사용할 PVC 정의의 예가 포함되어 있습니다. 을 참조하십시오 Trident 볼륨과 관련된 매개 변수 및 설정에 대한 자세한 설명을 확인하십시오.

쿠버네티스 `PersistentVolume` 오브젝트

Kubernetes "PersistentVolume" 개체는 Kubernetes 클러스터에서 사용할 수 있는 스토리지 부분을 나타냅니다. 사용 포드와 독립적인 라이프 사이클이 있습니다.

참고	Trident는 "PersistentVolume" 개체를 만들고 프로비저닝하는 볼륨을 기준으로 Kubernetes 클러스터에 자동으로 등록합니다. 스스로 관리할 수 없습니다.

Trident 기반의 'torageClass'를 참조하는 PVC를 생성하면 Trident는 해당 스토리지 클래스를 사용하여 새 볼륨을 프로비저닝하고 해당 볼륨에 대한 새 PV를 등록합니다. 프로비저닝 볼륨과 해당 PV를 구성할 때 Trident는 다음 규칙을 따릅니다.

- Trident는 Kubernetes의 PV 이름과 스토리지 프로비저닝에 사용되는 내부 이름을 생성합니다. 두 경우 모두 이름은 해당 범위에서 고유합니다.
- 볼륨의 크기는 플랫폼에 따라 가장 가까운 할당 가능한 수량으로 반올림될 수 있지만 PVC에서 요청된 크기와 최대한 가깝게 일치합니다.

쿠버네티스 `StorageClass` 오브젝트

Kubernetes의 torageClass 객체는 속성 세트를 사용하여 스토리지를 프로비저닝하기 위해 PersistentVolumeClaims의 이름으로 지정됩니다. 스토리지 클래스 자체는 사용할 구축 소유자를 식별하고 프로비저닝이 이해할 수 있는 조건으로 해당 자산 세트를 정의합니다.

관리자가 만들고 관리해야 하는 두 가지 기본 개체 중 하나입니다. 다른 하나는 Trident 백엔드 객체입니다.

Trident를 사용하는 Kubernetes의 torageClass 개체는 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 매개 변수는 Trident에만 해당되며 Trident에 클래스에 볼륨을 프로비저닝하는 방법을 알려줍니다.

스토리지 클래스 매개 변수는 다음과 같습니다.

속성	유형	필수 요소입니다	설명
속성	[string] 문자열을 매핑합니다	아니요	아래의 특성 섹션을 참조하십시오
스토리지 풀	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑
추가 StoragePools	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑
excludeStoragePools를 참조하십시오	Map [string] StringList 입니다	아니요	내의 스토리지 풀 목록에 백엔드 이름 매핑

스토리지 속성 및 가능한 값은 스토리지 풀 선택 특성 및 Kubernetes 속성으로 분류할 수 있습니다.

스토리지 풀 선택 특성입니다

이러한 매개 변수는 지정된 유형의 볼륨을 프로비저닝하는 데 사용해야 하는 Trident 관리 스토리지 풀을 결정합니다.

속성	유형	값	제공합니다	요청하십시오	에 의해 지원됩니다
미디어 [1]	문자열	HDD, 하이브리드, SSD	풀에는 이 유형의 미디어가 포함되어 있으며, 하이브리드는 둘 모두를 의미합니다	지정된 미디어 유형입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, solidfire-SAN
프로비저닝 유형	문자열	얇고 두껍습니다	풀은 이 프로비저닝 방법을 지원합니다	프로비저닝 방법이 지정되었습니다	Thick: All ONTAP; Thin: All ONTAP & solidfire-SAN
백엔드 유형	문자열	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	풀이 이 백엔드 유형에 속합니다	백엔드가 지정되었습니다	모든 드라이버
스냅샷 수	불입니다	참, 거짓	풀은 스냅샷이 있는 볼륨을 지원합니다	스냅샷이 활성화된 볼륨	온탑나스, 온탑씨, 솔리드파이어씨
복제	불입니다	참, 거짓	풀은 볼륨 클론을 지원합니다	클론이 활성화된 볼륨	온탑나스, 온탑씨, 솔리드파이어씨
암호화	불입니다	참, 거짓	풀은 암호화된 볼륨을 지원합니다	암호화가 활성화된 볼륨입니다	ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroups, ONTAP-SAN
IOPS	내부	양의 정수입니다	풀은 이 범위에서 IOPS를 보장할 수 있습니다	볼륨은 이러한 IOPS를 보장합니다	solidfire-SAN

[1]: ONTAP Select 시스템에서 지원되지 않습니다

대부분의 경우 요청된 값이 프로비저닝에 직접적인 영향을 미치며, 예를 들어 일반 프로비저닝을 요청하면 볼륨이 걸쭉하게 프로비저닝됩니다. 하지만 Element 스토리지 풀은 제공된 IOPS 최소 및 최대값을 사용하여 요청된 값이 아닌 QoS 값을 설정합니다. 이 경우 요청된 값은 스토리지 풀을 선택하는 데만 사용됩니다.

이상적으로는 '속성'을 단독으로 사용하여 특정 클래스의 요구 사항을 충족하는 데 필요한 스토리지의 품질을 모델링할 수 있습니다. Trident는 사용자가 지정한 '속성'의 _ALL_과 일치하는 스토리지 풀을 자동으로 검색하여 선택합니다.

클래스에 맞는 풀을 자동으로 선택하기 위해 속성(attributes)을 사용할 수 없는 경우, 'toragePools' 및 'additionalStoragePools' 매개 변수를 사용하여 풀을 더 세분화하거나 특정 풀 세트를 선택할 수도 있습니다.

'toragePools' 매개 변수를 사용하면 지정된 'attributes'와 일치하는 풀 세트를 추가로 제한할 수 있습니다. 즉, Trident는

프로비저닝에서 'attributes'와 'toragePools' 매개 변수로 식별되는 풀의 교집합을 사용합니다. 매개 변수만 사용하거나 둘 다 함께 사용할 수 있습니다.

"additionalStoragePools" 매개 변수를 사용하면 "attributes" 및 "toragePools" 매개 변수로 선택한 풀에 관계없이 Trident에서 프로비저닝에 사용하는 풀 집합을 확장할 수 있습니다.

'excludeStoragePools' 매개 변수를 사용하여 Trident에서 프로비저닝을 위해 사용하는 풀 집합을 필터링할 수 있습니다. 이 매개 변수를 사용하면 일치하는 풀이 모두 제거됩니다.

'toragePools' 및 'additionalStoragePools' 매개 변수에서 각 항목은 '<backend>:<storagePoolList>' 형식을 사용합니다. 여기서 '<storagePoolList>'는 지정된 백엔드에 대한 쉼표로 구분된 스토리지 풀 목록입니다. 예를 들어, additionalStoragePools 값은 ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze처럼 보일 수 있습니다. 이러한 목록에는 백엔드 및 목록 값 모두에 대한 regex 값이 적용됩니다. tridentctl 백엔드 가져오기 를 사용하여 백엔드와 해당 풀의 목록을 가져올 수 있습니다.

Kubernetes 특성

이러한 특성은 동적 프로비저닝 중 Trident가 스토리지 풀/백엔드를 선택하는 데 아무런 영향을 주지 않습니다. 대신 이러한 특성은 Kubernetes 영구 볼륨에서 지원하는 매개 변수만 제공합니다. 작업자 노드는 파일 시스템 생성 작업을 담당하며 xfsprogs와 같은 파일 시스템 유틸리티가 필요할 수 있습니다.

속성	유형	값	설명	관련 드라이버	Kubernetes 버전
fsType입니다	문자열	ext4, ext3, xfs	블록 볼륨의 파일 시스템 유형입니다	solidfire-SAN, ONTAP-NAS, ONTAP-NAS-이코노미, ONTAP-NAS-Flexgroup, ONTAP-SAN, ONTAP-SAN - 경제성	모두
allowVolumeExpansion	부울	참, 거짓	PVC 크기 증가에 대한 지원을 활성화 또는 비활성화합니다	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
볼륨BindingMode를 선택합니다	문자열	Immediate, WaitForFirstConsumer입니다	볼륨 바인딩 및 동적 프로비저닝이 수행될 시기를 선택합니다	모두	1.19-1.26

팁

- 를 클릭합니다 `fsType` 매개 변수는 SAN LUN에 대해 원하는 파일 시스템 유형을 제어하는 데 사용됩니다. 또한 Kubernetes는 의 존재 여부를 사용합니다 `fsType` 파일 시스템이 있음을 나타내는 스토리지 클래스에 있습니다. 볼륨 소유권은 를 사용하여 제어할 수 있습니다 `fsGroup` POD의 보안 컨텍스트는 에만 해당됩니다 `fsType` 가 설정됩니다. 을 참조하십시오 "Kubernetes: Pod 또는 컨테이너의 보안 컨텍스트를 구성합니다" 를 사용하여 볼륨 소유권을 설정하는 방법에

대한 개요를 보려면 를 참조하십시오 `fsGroup` 상황. Kubernetes가 에 적용됩니다 `fsGroup` 다음 경우에만 값:

- 스토리지 클래스에 fsType이 설정되어 있습니다.
- PVC 액세스 모드는 RWO입니다.

NFS 스토리지 드라이버의 경우 파일 시스템이 NFS 내보내기의 일부로 이미 존재합니다. fsGroup을 사용하려면 스토리지 클래스가 여전히 fsType을 지정해야 합니다. NFS 또는 null이 아닌 값으로 설정할 수 있습니다.

- 을 참조하십시오 "볼륨 확장" 볼륨 확장에 대한 자세한 내용은 를 참조하십시오.
- Trident 설치 프로그램 번들에는 'Sample-input/storage-class- *.YAML'의 Trident와 함께 사용할 수 있는 여러 가지 스토리지 클래스 정의가 포함되어 있습니다. Kubernetes 스토리지 클래스를 삭제하면 해당 Trident 스토리지 클래지도 삭제됩니다.

쿠버네티스 `VolumeSnapshotClass` 오브젝트

쿠버네티스 VolumeSnapshotClass 객체는 'torageClaes'와 유사합니다. 이 기능을 사용하면 여러 스토리지 클래스를 정의할 수 있으며, 스냅샷을 필요한 스냅샷 클래스와 연결하기 위해 볼륨 스냅숏에서 참조할 수 있습니다. 각 볼륨 스냅샷은 단일 볼륨 스냅샷 클래스와 연결됩니다.

스냅샷을 생성하려면 관리자가 VolumeSnapshotClass를 정의해야 합니다. 볼륨 스냅샷 클래스는 다음과 같은 정의로 생성됩니다.

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

driver는 CSI-snapclass 클래스의 볼륨 스냅샷을 요청하는 Kubernetes를 Trident에서 처리하도록 지정합니다. "eletionPolicy"는 스냅샷을 삭제할 때 수행할 작업을 지정합니다. "eletionPolicy"를 "Delete"로 설정하면 스냅샷이 삭제될 때 스토리지 클러스터의 기본 스냅샷과 볼륨 스냅샷 객체가 제거됩니다. 또는 '유지'로 설정하면 VolumeSnapshotContent와 물리적 스냅샷이 보존됩니다.

쿠버네티스 `VolumeSnapshot` 오브젝트

Kubernetes 'VolumeSnapshot' 개체는 볼륨의 스냅샷을 생성하는 요청입니다. PVC는 사용자가 볼륨에 대해 요청하는 것처럼 볼륨 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성하도록 요청하는 것입니다.

볼륨 스냅샷 요청이 들어오면 Trident는 백엔드의 볼륨에 대한 스냅샷 생성을 자동으로 관리하고 고유한 'VolumeSnapshotContent' 객체를 생성하여 스냅샷을 표시합니다. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 스냅샷을 DataSource로 사용할 수 있습니다.

참고 VolumeSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident는 이 PVC에 대한 백업 볼륨을 * Deleting * 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 볼륨이 제거됩니다.

쿠버네티스 `VolumeSnapshotContent` 오브젝트

Kubernetes의 'VolumeSnapshotContent' 객체는 이미 프로비저닝된 볼륨에서 생성된 스냅샷을 나타냅니다. 이 스냅샷은 "PersistentVolume"과 유사하며 스토리지 클러스터에서 프로비저닝된 스냅샷을 나타냅니다. 스냅샷이 생성될 때 PersistentVolumeClaim 및 PersistentVolume 개체와 마찬가지로 VolumeSnapshotContent 개체는 스냅샷 생성을 요청한 VolumeSnapshot 객체에 대한 일대일 매핑을 유지합니다.

VolumeSnapshotContent 객체에는 스냅샷 스냅샷(스냅샷 핸들 등)을 고유하게 식별하는 세부 정보가 포함되어 있습니다. 이 나프산Handle은 PV의 이름과 VolumeSnapshotContent 객체의 이름을 조합한 독특한 것이다.

스냅샷 요청이 들어오면 Trident가 백엔드에 스냅샷을 생성합니다. 스냅샷이 생성된 후 Trident는 'VolumeSnapshotContent' 객체를 구성하여 해당 스냅샷을 Kubernetes API에 노출합니다.

참고	일반적으로 오브젝트를 관리할 필요가 `VolumeSnapshotContent` 없습니다. Trident 외부에서 만들려는 경우는 "볼륨 스냅샷을 가져옵니다"예외입니다.

Kubernetes `VolumeGroupSnapshotClass` 오브젝트

Kubernetes `VolumeGroupSnapshotClass` 오브젝트는 과 `VolumeSnapshotClass`유사합니다. 이러한 스냅샷은 여러 스토리지 클래스를 정의하는 데 도움이 되며, 볼륨 그룹 스냅샷에서 참조되어 스냅샷을 필요한 스냅샷 클래스와 연결합니다. 각 볼륨 그룹 스냅샷은 단일 볼륨 그룹 스냅샷 클래스와 연결됩니다.

에이 `VolumeGroupSnapshotClass` 스냅샷 그룹을 생성하려면 관리자가 정의해야 합니다. 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
```

는 Trident에서 처리합니다. `deletionPolicy` 그룹 스냅샷을 삭제해야 할 때 수행할 작업을 지정합니다 . `deletionPolicy` 로 설정됩니다 `Delete` 스냅샷이 삭제되면 볼륨 그룹 스냅샷 개체와 스토리지 클러스터의 기본 스냅샷도 제거됩니다. 또는 로 설정하면 `Retain` `VolumeGroupSnapshotContent` 및 물리적 스냅샷이 보존됩니다.

Kubernetes `VolumeGroupSnapshot` 오브젝트

쿠버네티스 `VolumeGroupSnapshot` 객체는 여러 볼륨의 스냅샷을 생성하라는 요청입니다. PVC가 사용자가 볼륨에 대해 요청한 것을 나타내는 것처럼, 볼륨 그룹 스냅샷은 사용자가 기존 PVC의 스냅샷을 생성하라는 요청입니다.

볼륨 그룹 스냅샷 요청이 들어오면 Trident는 백엔드의 볼륨에 대한 그룹 스냅샷 생성을 자동으로 관리하고 고유한 스냅샷을 생성하여 스냅샷을 노출합니다. `VolumeGroupSnapshotContent` 물체. 기존 PVC에서 스냅샷을 생성하고 새 PVC를 생성할 때 스냅샷을 DataSource로 사용할 수 있습니다.

참고	VolumeGroupSnapshot의 수명 주기는 소스 PVC와 무관합니다. 즉, 소스 PVC가 삭제된 후에도 스냅샷은 유지됩니다. 연관된 스냅샷이 있는 PVC를 삭제할 때 Trident는 이 PVC에 대한 백업 볼륨을 * Deleting * 상태로 표시하지만 완전히 제거하지는 않습니다. 연결된 모든 스냅샷이 삭제되면 볼륨 그룹 스냅샷도 제거됩니다.

Kubernetes `VolumeGroupSnapshotContent` 오브젝트

쿠버네티스 `VolumeGroupSnapshotContent` 개체는 이미 프로비저닝된 볼륨에서 가져온 그룹 스냅샷을 나타냅니다. 이 스냅샷은 A와 유사하며 `PersistentVolume` 스토리지 클러스터에서 프로비저닝된 스냅샷을 나타냅니다. 및 `PersistentVolume` 객체와 마찬가지로 `PersistentVolumeClaim` 스냅샷이 생성될 때 `VolumeSnapshotContent` 객체는 스냅샷 생성을 요청한 객체에 대한 일대일 매핑을 `VolumeSnapshot` 유지합니다.

그만큼 `VolumeGroupSnapshotContent` 개체에는 스냅샷 그룹을 식별하는 세부 정보(예: `volumeGroupSnapshotHandle` 그리고 개별 볼륨 스냅샷이 스토리지 시스템에 존재합니다.

스냅샷 요청이 들어오면 Trident는 백엔드에 볼륨 그룹 스냅샷을 생성합니다. 볼륨 그룹 스냅샷이 생성된 후 Trident는 `VolumeGroupSnapshotContent` 객체를 생성하여 Kubernetes API에 스냅샷을 노출합니다.

쿠버네티스 `CustomResourceDefinition` 오브젝트

Kubernetes 사용자 지정 리소스는 관리자가 정의하며 비슷한 객체를 그룹화하는 데 사용되는 Kubernetes API의 엔드포인트입니다. Kubernetes에서는 오브젝트 컬렉션을 저장하기 위한 사용자 지정 리소스의 생성을 지원합니다. kubeck Get CRD를 실행하여 이러한 리소스 정의를 얻을 수 있습니다.

사용자 정의 리소스 정의(CRD) 및 관련 오브젝트 메타데이터는 Kubernetes에서 메타데이터 저장소에 저장됩니다. 따라서 Trident를 위한 별도의 저장소가 필요하지 않습니다.

Trident에서는 오브젝트를 사용하여 `CustomResourceDefinition` Trident 백 엔드, Trident 스토리지 클래스 및 Trident 볼륨과 같은 Trident 오브젝트의 ID를 유지합니다. 이러한 오브젝트는 Trident에서 관리합니다. 또한 CSI 볼륨 스냅샷 프레임워크는 볼륨 스냅샷을 정의하는 데 필요한 일부 CRD를 소개합니다.

CRD는 Kubernetes를 구성하는 것입니다. 위에 정의된 리소스의 객체는 Trident에 의해 생성됩니다. 간단히 예로, 'tridentctl'을 사용하여 백엔드를 생성할 때 해당하는 'tridentbackends' CRD 객체는 Kubernetes에서 사용할 수 있도록 생성됩니다.

다음은 Trident의 CRD에 대해 고려해야 할 몇 가지 사항입니다.

- Trident가 설치되면 일련의 CRD가 생성되어 다른 리소스 유형과 마찬가지로 사용할 수 있습니다.
- 를 사용하여 Trident를 제거하는 경우 `tridentctl uninstall` Command, Trident Pod가 삭제되지만 생성된 CRD는 정리되지 않습니다. 을 참조하십시오 "Trident를 제거합니다" Trident를 완전히 제거하고 처음부터 다시 구성할 수 있는 방법을 이해합니다.

Trident 개체 `StorageClass`

Trident가 Kubernetes에 맞는 스토리지 클래스를 생성합니다 `StorageClass` 지정하는 개체입니다 `csi.trident.netapp.io` 그들의 공급자 분야. 스토리지 클래스 이름이 Kubernetes의 클래스 이름과 일치합니다 `StorageClass` 나타내는 개체입니다.

참고	Kubernetes를 사용하면 Trident를 프로비저닝한 Kubernetes의 torageClass가 등록되면 이러한

객체가 자동으로 생성됩니다.

스토리지 클래스는 볼륨에 대한 일련의 요구 사항으로 구성됩니다. Trident는 이러한 요구 사항을 각 스토리지 풀에 있는 속성과 일치시킵니다. 일치하는 경우 해당 스토리지 풀이 해당 스토리지 클래스를 사용하여 볼륨을 프로비저닝할 수 있는 유효한 타겟입니다.

REST API를 사용하여 스토리지 클래스를 직접 정의하는 스토리지 클래스 구성을 생성할 수 있습니다. 그러나 Kubernetes 구축의 경우 새로운 Kubernetes의 torageClass 오브젝트를 등록할 때 이러한 객체가 생성되기를 기대합니다.

Trident 백엔드 객체

백엔드는 Trident가 볼륨을 프로비저닝하는 스토리지 공급자를 나타냅니다. 단일 Trident 인스턴스가 원하는 수의 백엔드를 관리할 수 있습니다.

참고 | 이것은 직접 만들고 관리하는 두 가지 개체 유형 중 하나입니다. 다른 하나는 Kubernetes의 torageClass 오브젝트입니다.

이러한 개체를 구성하는 방법에 대한 자세한 내용은 을 참조하십시오 "백엔드 구성 중".

Trident 개체 `StoragePool`

스토리지 풀은 각 백엔드에서 프로비저닝에 사용할 수 있는 고유한 위치를 나타냅니다. ONTAP 의 경우 이는 SVM의 집계에 해당합니다. NetApp HCI/ SolidFire 의 경우 이는 관리자가 지정한 QoS 대역에 해당합니다. 각 스토리지 풀에는 성능 특성과 데이터 보호 특성을 정의하는 일련의 고유한 스토리지 속성이 있습니다.

다른 오브젝트와 달리 스토리지 풀 후보 는 항상 자동으로 검색되고 관리됩니다.

Trident 개체 `Volume`

볼륨은 프로비저닝의 기본 단위로, NFS 공유, iSCSI 및 FC LUN 등의 백엔드 엔드포인트를 구성합니다. Kubernetes에서 이러한 항목은 에 직접 `PersistentVolumes` 대응합니다. 볼륨을 생성할 때 볼륨의 용량을 할당할 수 있는 위치와 크기를 결정하는 스토리지 클래스가 있는지 확인합니다.

참고 |
- Kubernetes에서 이러한 오브젝트는 자동으로 관리됩니다. 프로비저닝 Trident를 보려면 해당 Trident를 확인하십시오.
- 연결된 스냅샷이 있는 PV를 삭제하면 해당 Trident 볼륨이 * Deleting * 상태로 업데이트됩니다. Trident 볼륨을 삭제하려면 볼륨의 스냅샷을 제거해야 합니다.

볼륨 구성은 프로비저닝된 볼륨에 있어야 하는 속성을 정의합니다.

속성	유형	필수 요소입니다	설명
버전	문자열	아니요	Trident API 버전("1")
이름	문자열	예	생성할 볼륨의 이름입니다
storageClass 를 선택합니다	문자열	예	볼륨을 프로비저닝할 때 사용할 스토리지 클래스입니다

속성	유형	필수 요소입니다	설명
크기	문자열	예	용량 할당할 볼륨의 크기(바이트)입니다
프로토콜	문자열	아니요	사용할 프로토콜 유형;"파일" 또는 "블록"
내부 이름	문자열	아니요	스토리지 시스템에 있는 객체의 이름으로, Trident에서 생성
CloneSourceVolume	문자열	아니요	ONTAP(NAS, SAN) 및 SolidFire - *: 복제할 볼륨의 이름입니다
SplitOnClone 을 참조하십시오	문자열	아니요	ONTAP(NAS, SAN): 상위 클론에서 클론을 분할합니다
스냅샷 정책	문자열	아니요	ONTAP - *: 사용할 스냅샷 정책
snapshotReserve	문자열	아니요	ONTAP - *: 스냅숏용으로 예약된 볼륨의 비율입니다
내보내기 정책	문자열	아니요	ONTAP-NAS *: 사용할 엑스포트 정책
스냅샷 디렉토리	불입니다	아니요	ONTAP-NAS *: 스냅샷 디렉토리가 표시되는지 여부를 나타냅니다
unixPermissions	문자열	아니요	ONTAP-NAS *: 초기 UNIX 권한
블록 크기	문자열	아니요	SolidFire - *: 블록/섹터 크기
파일 시스템	문자열	아니요	파일 시스템 유형입니다
skipRecoveryQueue	문자열	아니요	볼륨을 삭제하는 동안 저장소의 복구 대기열을 우회하고 볼륨을 즉시 삭제합니다.

Trident는 볼륨을 생성할 때 'internalName'을 생성합니다. 이 단계는 두 단계로 구성됩니다. 먼저, 저장소 접두사(기본 "트리덴트" 또는 백엔드 구성의 접두사)를 볼륨 이름에 추가하여 "<prefix>-<volume-name>" 형식의 이름을 만듭니다. 그런 다음 백엔드에서 허용되지 않는 문자를 대체하여 이름을 삭제하는 작업을 진행합니다. ONTAP 백엔드의 경우 하이픈을 밑줄로 바꿉니다. 따라서 내부 이름은 "<prefix>_<volume-name>"이 됩니다. 요소 백엔드의 경우 밑줄을 하이픈으로 바꿉니다.

볼륨 구성을 사용하여 REST API를 사용하여 볼륨을 직접 프로비저닝할 수 있지만 Kubernetes 배포에서는 대부분의 사용자가 표준 Kubernetes "PersistentVolumeClaim" 방법을 사용할 것으로 예상됩니다. Trident는 프로비저닝 프로세스의 일부로 이 볼륨 개체를 자동으로 만듭니다.

Trident 개체 `Snapshot`

스냅샷은 볼륨의 시점 복제본으로, 새 볼륨을 용량 할당하거나 복구 상태를 복구하는 데 사용할 수 있습니다. Kubernetes에서는 이러한 객체가 'VolumeSnapshotContent' 객체와 직접 일치합니다. 각 스냅샷은 스냅샷에 대한

데이터의 소스인 볼륨에 연결됩니다.

각 '스냅샷' 개체에는 아래 나열된 속성이 포함됩니다.

속성	유형	필수 요소입니다	설명
버전	문자열	예	Trident API 버전("1")
이름	문자열	예	Trident 스냅샷 개체의 이름입니다
내부 이름	문자열	예	스토리지 시스템의 Trident 스냅샷 개체의 이름입니다
볼륨 이름	문자열	예	스냅샷이 생성된 영구 볼륨의 이름입니다
볼륨 국제 이름	문자열	예	스토리지 시스템에서 연결된 Trident 볼륨 개체의 이름입니다

참고 Kubernetes에서 이러한 오브젝트는 자동으로 관리됩니다. 프로비저닝 Trident를 보려면 해당 Trident를 확인하십시오.

Kubernetes 'VolumeSnapshot' 객체 요청이 생성되면 Trident는 백업 스토리지 시스템에 스냅샷 객체를 생성하여 작동합니다. 이 스냅샷 개체의 인터날Name은 볼륨 스냅샷 개체의 UID(예: 스냅샷-e8d8a0ca-9826-11e9-9807-525400f3f660)와 접두사 스냅샷-UID를 결합하여 생성됩니다. 볼륨 이름 과 볼륨 InternalName 은 백업 볼륨의 세부 정보를 가져오는 방식으로 채워집니다.

Trident 개체입니다 `ResourceQuota`

Trident deamonset은 `system-node-critical` Trident가 정상적인 노드 종료 중에 볼륨을 식별 및 정리하고 Trident demonset Pod가 리소스 압력이 높은 클러스터에서 낮은 우선 순위로 워크로드를 사전 지정할 수 있도록 하기 위해 Kubernetes에서 사용 가능한 가장 높은 우선 순위 클래스인 우선 순위 클래스를 사용합니다.

이를 위해 Trident는 객체를 사용하여 `ResourceQuota` Trident 데몬 세트의 "시스템 노드 중요" 우선 순위 클래스가 충족되도록 합니다. 배포 및 demonset 생성 전에 Trident는 객체를 찾고 `ResourceQuota` 발견되지 않은 경우 이를 적용합니다.

기본 리소스 할당량 및 우선순위 클래스에 대한 더 많은 제어가 필요한 경우 'CUSTOM.YAML'을 생성하거나 제어 차트를 사용하여 'ResourceQuota' 객체를 구성할 수 있습니다.

다음은 Trident 데모의 우선 순위를 지정하는 'ResourceQuota' 개체의 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 을 참조하십시오 "Kubernetes: 리소스 할당량".

정리 `ResourceQuota` 설치에 실패한 경우

드문 경우지만 ResourceQuota 객체를 만든 후 설치가 실패하는 경우 먼저 시도해 보십시오 "제거 중" 그런 다음 다시 설치합니다.

이 기능이 작동하지 않으면 수동으로 ResourceQuota 객체를 제거합니다.

제거 `ResourceQuota`

리소스 할당을 직접 제어하려는 경우 다음 명령을 사용하여 Trident 개체를 제거할 수 `ResourceQuota` 있습니다.

```
kubectl delete quota trident-csi -n trident
```

POD 보안 표준(PSS) 및 보안 컨텍스트 제약(SCC)

Kubernetes Pod 보안 표준(PSS) 및 Pod 보안 정책(PSP)에서 사용 권한 수준을 정의하고 Pod의 동작을 제한합니다. OpenShift Security Context Constraints(SCC)도 OpenShift Kubernetes Engine에 특정한 POD 제한을 정의합니다. 이러한 사용자 지정 기능을 제공하기 위해 Trident는 설치 중에 특정 권한을 활성화합니다. 다음 섹션에서는 Trident에서 설정한 사용 권한에 대해 자세히 설명합니다.

> 참고 PSS는 Pod 보안 정책(PSP)을 대체합니다. PSP는 Kubernetes v1.21에서 사용되지 않으며 v1.25에서 제거됩니다. 자세한 내용은 을 참조하십시오 "Kubernetes: 보안".

필요한 Kubernetes 보안 컨텍스트 및 관련 필드

권한	설명
특별 권한	CSI를 사용하려면 마운트 지점이 양방향이어야 합니다. 즉, Trident 노드 포드가 권한이 있는 컨테이너를 실행해야 합니다. 자세한 내용은 을 참조하십시오 "Kubernetes: 마운트 전파".
호스트 네트워킹	iSCSI 데몬에 필요합니다. iscsiadm은 iSCSI 마운트를 관리하고 호스트 네트워킹을 사용하여 iSCSI 데몬과 통신합니다.
호스트 IPC	NFS는 IPC(프로세스 간 통신)를 사용하여 NFSD와 통신합니다.

권한	설명
호스트 PID	NFS를 시작하기 위해 `rpc-statd` 필요합니다. Trident는 NFS 볼륨을 마운트하기 전에 호스트 프로세스를 쿼리하여 가 실행 중인지 여부를 `rpc-statd` 확인합니다.
제공합니다	'SYS_ADMIN' 기능은 권한 있는 컨테이너의 기본 기능의 일부로 제공됩니다. 예를 들어, Docker는 권한이 있는 컨테이너인 'CapPrm:0000003fffffffffff'CapEff:0000003fffffffffff'에 대해 이러한 기능을 설정합니다
Seccomp	Seccomp 프로필은 권한이 있는 컨테이너에서 항상 "제한 없음"이므로 Trident에서 활성화할 수 없습니다.
SELinux	OpenShift에서는 권한이 있는 컨테이너가 `spc_t` ("상위 권한 있는 컨테이너") 도메인에서 실행되고 권한이 없는 컨테이너는 `container_t` 도메인에서 실행됩니다. `containerd`가 설치되어 있으면 `container-selinux` 모든 컨테이너가 도메인에서 실행되어 `spc_t` SELinux가 효과적으로 비활성화됩니다. 따라서 Trident는 컨테이너에 추가되지 `seLinuxOptions` 않습니다.
DAC	권한이 있는 컨테이너는 루트로 실행되어야 합니다. 권한이 없는 컨테이너는 root로 실행되어 CSI에 필요한 UNIX 소켓에 액세스합니다.

POD 보안 표준(PSS)

라벨	설명	기본값
pod-security.Kubernetes.io/force 포드-security.Kubernetes.io/force -version을 적용합니다	Trident 컨트롤러와 노드를 설치 네임스페이스에 받아들일 수 있습니다. 네임스페이스 레이블을 변경하지 마십시오.	최근 실시한 PSS의 최신 버전이나 PSS의 최고 버전 >

경고 네임스페이스 레이블을 변경하면 포드가 예약되지 않고 "오류 생성:..." 또는 "경고: 트리덴트 - CSI -..." 가 발생할 수 있습니다. 이 경우 특권 네임스페이스 레이블이 변경되었는지 확인합니다. 있는 경우 Trident를 다시 설치합니다.

PSP(POD 보안 정책)

필드에 입력합니다	설명	기본값
'allowPrivilegeegeEscalation'	권한 있는 컨테이너는 권한 에스컬레이션을 허용해야 합니다.	"참"입니다
'allowedCSIDrivers'입니다	Trident는 인라인 CSI 임시 볼륨을 사용하지 않습니다.	비어 있습니다
'allowedCapabilities'	권한이 없는 Trident 컨테이너는 기본 세트보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너에 모든 가능한 기능이 부여됩니다.	비어 있습니다

필드에 입력합니다	설명	기본값
'allowedFlexVolumes'	Trident는 을 사용하지 않습니다 "FlexVolume 드라이버"따라서 허용된 볼륨 목록에 포함되지 않습니다.	비어 있습니다
'allowedHostPaths'	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트하므로 이 목록을 설정하는 데는 아무런 이점이 없습니다.	비어 있습니다
'allowedProcMountTypes'	Trident는 'ProcMountTypes'를 사용하지 않습니다.	비어 있습니다
'allowedUnsafeSysctls'	Trident는 안전하지 않은 '스컬'을 필요로 하지 않습니다.	비어 있습니다
"기본 추가 기능"을 참조하십시오	권한이 있는 컨테이너에 기능을 추가할 필요가 없습니다.	비어 있습니다
월권 AllowPrivilegeEscalation	권한 에스컬레이션을 허용하는 작업은 각 Trident 포드에서 처리됩니다.	거짓입니다
'바이디니데시스틀스'	'스매스들'은 허용되지 않습니다.	비어 있습니다
'fsGroup'입니다	Trident 컨테이너가 루트로 실행됩니다.	러아사니
호스티IPC	NFS 볼륨을 마운트하려면 호스트 IPC가 nfsd와 통신해야 합니다	"참"입니다
호스트 네트워크	iscsiadm을 사용하려면 호스트 네트워크가 iSCSI 데몬과 통신해야 합니다.	"참"입니다
'hostPID'	노드에서 RPC-statd가 실행되고 있는지 확인하려면 호스트 PID가 필요합니다.	"참"입니다
호스트 포트	Trident는 호스트 포트를 사용하지 않습니다.	비어 있습니다
특권	Trident 노드 포드는 볼륨을 마운트하려면 권한이 있는 컨테이너를 실행해야 합니다.	"참"입니다
"RootFilesystem"을 선택합니다	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	거짓입니다
레퀴레드드롭카포비스	Trident 노드 포드는 권한이 있는 컨테이너를 실행하고 기능을 삭제할 수 없습니다.	"없음"
루아그룹	Trident 컨테이너가 루트로 실행됩니다.	러아사니
'runAsUser'입니다	Trident 컨테이너가 루트로 실행됩니다.	루아안니

필드에 입력합니다	설명	기본값
'런타임 클래스'	트라이던트(Trident)는 RuntimeClasses를 사용하지 않습니다.	비어 있습니다
'e linux'	Trident는 현재 컨테이너 실행 시간과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있으므로 'eLinuxOptions'를 설정하지 않습니다.	비어 있습니다
업플레탈그룹	Trident 컨테이너가 루트로 실행됩니다.	러아사니
'볼륨'	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	대스트패스, 투상도, 최고가

SCC(Security Context Constraints)

라벨	설명	기본값
'allowHostDirVolumePlugin'을 선택합니다	Trident 노드 포드는 노드의 루트 파일 시스템을 마운트합니다.	"참"입니다
'allowHostIPC'입니다	NFS 볼륨을 마운트하려면 호스트 IPC가 nfsd와 통신해야 합니다.	"참"입니다
'allowHostNetwork'입니다	iscsiadm을 사용하려면 호스트 네트워크가 iSCSI 데몬과 통신해야 합니다.	"참"입니다
'allowHostPID'	노드에서 RPC-statd가 실행되고 있는지 확인하려면 호스트 PID가 필요합니다.	"참"입니다
'allowHostPorts'입니다	Trident는 호스트 포트를 사용하지 않습니다.	거짓입니다
'allowPrivilegeegeEscalation'	권한 있는 컨테이너는 권한 에스컬레이션을 허용해야 합니다.	"참"입니다
'allowPrivilegedContainer'	Trident 노드 포드는 볼륨을 마운트하려면 권한이 있는 컨테이너를 실행해야 합니다.	"참"입니다
'allowedUnsafeSysctls'	Trident는 안전하지 않은 '스컬'을 필요로 하지 않습니다.	"없음"
'allowedCapabilities'	권한이 없는 Trident 컨테이너는 기본 세트보다 더 많은 기능을 필요로 하지 않으며 권한이 있는 컨테이너에 모든 가능한 기능이 부여됩니다.	비어 있습니다
"기본 추가 기능"을 참조하십시오	권한이 있는 컨테이너에 기능을 추가할 필요가 없습니다.	비어 있습니다
'fsGroup'입니다	Trident 컨테이너가 루트로 실행됩니다.	러아사니

라벨	설명	기본값
그룹	이 SCC는 Trident에만 해당되며 사용자에게 바인딩됩니다.	비어 있습니다
"RootFilesystem"을 선택합니다	Trident 노드 포드는 노드 파일 시스템에 써야 합니다.	거짓입니다
레퀴레드드롭카포비스	Trident 노드 포드는 권한이 있는 컨테이너를 실행하고 기능을 삭제할 수 없습니다.	"없음"
'runAsUser’입니다	Trident 컨테이너가 루트로 실행됩니다.	러아사니
새리눅스컨텍스트	Trident는 현재 컨테이너 실행 시간과 Kubernetes 배포판이 SELinux를 처리하는 방식에 차이가 있으므로 'eLinuxOptions’를 설정하지 않습니다.	비어 있습니다
'eccompProfiles	특권 컨테이너는 항상 "비제한" 상태로 실행됩니다.	비어 있습니다
업플레탈그룹	Trident 컨테이너가 루트로 실행됩니다.	러아사니
'사용자'	이 SCC를 Trident 네임스페이스의 Trident 사용자에게 바인딩하기 위해 하나의 항목이 제공됩니다.	해당 없음
'볼륨'	Trident Pod에는 이러한 볼륨 플러그인이 필요합니다.	hostPath, downwardAPI, 투영, emptyDir

법적 고지

법적 고지 사항은 저작권 선언, 상표, 특허 등에 대한 액세스를 제공합니다.

저작권

"https://www.netapp.com/company/legal/copyright/"

상표

NetApp, NetApp 로고, NetApp 상표 페이지에 나열된 마크는 NetApp Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.

"https://www.netapp.com/company/legal/trademarks/"

특허

NetApp 소유 특허 목록은 다음 사이트에서 확인할 수 있습니다.

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

개인 정보 보호 정책

"https://www.netapp.com/company/legal/privacy-policy/"

오픈 소스

Trident용 NetApp 소프트웨어에 사용된 타사 저작권 및 라이선스는 의 각 릴리스에 대한 고지 파일에서 검토할 수 있습니다 https://github.com/NetApp/trident/.

저작권 정보

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

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

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

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

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

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

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

상표 정보

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