



Astra Data Store 문서

Astra Data Store

NetApp
June 13, 2022

목차

Astra Data Store 문서	1
릴리스 정보	2
Astra Data Store 이번 릴리스의 새로운 기능	2
알려진 문제	3
알려진 제한 사항	3
개념	5
Astra Data Store에 대해 자세히 알아보십시오	5
Astra Data Store 배포 모델	6
클러스터 확장	7
Astra Data Store의 스토리지 효율성	8
Astra Data Store의 보안	8
시작하십시오	10
Astra 데이터 저장소 요구 사항	10
Astra Data Store를 빠르게 시작합니다	14
Astra Data Store를 설치합니다	16
Astra Data Store 구성 요소를 설정합니다	33
Astra Data Store 초기 액세스 프로그램(EAP) 릴리스 제한	44
Astra Data Store에 대한 질문과 대답	45
Astra Data Store를 사용하십시오	48
kubectx 명령을 사용하여 Astra Data Store 리소스를 관리합니다	48
테스트 응용 프로그램을 배포합니다	54
Astra Data Store 클러스터를 관리합니다	58
Astra Data Store를 모니터링합니다	68
보안 Astra 데이터 저장소	83
Astra Data Store 라이선스를 업데이트합니다	87
Astra Data Store를 업그레이드합니다	88
자동 스크립트로 Astra Data Store를 제거합니다	90
VMware와 함께 Astra Data Store를 사용하십시오	105
VMware를 사용한 Astra Data Store에 대해 자세히 알아보십시오	105
VMware 요구 사항이 있는 Astra Data Store	105
VMware와 함께 Astra Data Store를 설정합니다	106
VMware 설치의 구성 요소를 모니터링합니다	113
VMware 설치의 Astra Data Store 구성 요소를 관리합니다	115
VMware 통합 환경에서 Astra Data Store를 제거합니다	118
지식 및 지원	119
문제 해결	119
도움을 받으십시오	119
자동 지원 모니터링	119
이전 버전의 Astra Data Store	124

법적 고지 125

 저작권 125

 상표 125

 특허 125

 개인 정보 보호 정책 125

 오픈 소스 125

Astra Data Store 문서

릴리스 정보

Astra Data Store의 2022.05 조기 액세스 프로그램(EAP) 릴리스를 발표하게 되어 기쁘게 생각합니다.

- ["Astra Data Store 이번 릴리스의 새로운 기능"](#)
- ["알려진 문제"](#)
- ["알려진 제한 사항"](#)

Twitter @NetAppDoc을 팔로우하십시오. 가 되어 문서에 대한 피드백을 보냅니다 ["GitHub 기고자입니다"](#) 또는 doccomments@netapp.com 으로 이메일을 보내주십시오.

Astra Data Store 이번 릴리스의 새로운 기능

Astra Data Store의 2022.05 조기 액세스 프로그램(EAP) 릴리스를 발표하게 되어 기쁘게 생각합니다.

2022년 5월 31일(2022.05 EAP 릴리스)

Astra Data Store 2022.05 릴리즈에는 다음과 같은 기능이 포함되어 있습니다.

- VMware 환경 통합:
 - NFS VVOL 데이터 저장소 워크플로우 및 스토리지 정책 기반 관리 스토리지 프로비저닝
 - vCenter에서 기본 스토리지 관리 및 모니터링을 위한 VMware vSphere용 Astra 플러그인
- 클러스터 관리 기능이 향상되었습니다 ["Astra Control Center에서 GUI 기반 클러스터 구축"](#).
- 에 대한 지원 ["대규모 클러스터 확장"](#) (노드 수 증가, CPU 및 용량 증가)
- 보안 강화(키 및 인증서, 외부 키 관리(KMIP) 지원)

2022년 2월 8일(2021.12.1)

Astra Data Store 미리 보기(21.12)용 패치 릴리스(2021.12.1)입니다.

- 이 릴리즈에서는 Astra Data Store 미리 보기가 Calico CNI를 사용한 VXLAN 구성을 지원합니다.
- Calico WireGuard 지원 네트워크 구성은 이제 Astra Data Store Preview에서 지원됩니다.
- 업데이트된 AstraDSCluster.YAML 'adsNetworkInterfaces' 섹션에 보다 자세한 설명이 추가되었습니다.
- kubeck 명령 제거 프로세스 대신 새로운 Astra Data Store 제거 스크립트가 제공됩니다.

2021년 12월 21일(21.12)

Astra Data Store Preview 초기 릴리즈.

- ["그게 뭐죠"](#)
- ["구축 모델 및 구성 요소"](#)
- ["시작하는 데 필요한 사항"](#)
- ["설치합니다"](#) 및 ["설정"](#)

- "관리" 및 "모니터링" 성능
- "Cloud Insights를 사용한 모니터링"
- "도움을 받으십시오" 및 "자동 지원 모니터링을 사용합니다"

자세한 내용을 확인하십시오

- "알려진 문제"
- "알려진 제한 사항"
- "NetApp 기술 자료 문서"

알려진 문제

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

Astra Data Store 2202.05 조기 액세스 프로그램(EAP) 릴리스

이 릴리스에는 이러한 알려진 문제가 포함되어 있습니다. 이러한 알려진 문제를 주의 깊게 검토하십시오.

vSphere에 스토리지 백엔드 작업이 표시되지 않습니다

기존 스토리지 백엔드를 사용하거나 스토리지 백엔드를 제거하는 작업이 제출되면 vSphere Client의 Recent Tasks 패널에 작업이 나타나지 않습니다.

VM을 생성할 때 추가 폴더가 생성됩니다

vSphere에서 VM을 생성할 때 데이터 저장소 아래의 VM당 단일 폴더 대신 추가 폴더가 생성되는 경우가 있습니다.

VMware vSphere용 Astra Plugin의 필터링된 보기에는 데이터가 간헐적으로 표시되지 않습니다

가상 머신에 대한 테이블 열 필터는 필터링된 데이터를 생략하는 경우가 있습니다. 필터링된 보기에는 로드 아이콘이 대신 표시됩니다. 이 문제를 해결하려면 다른 보기로 이동하여 필터를 다시 설정하여 데이터를 다시 표시할 수 있습니다.

Kubernetes 작업자 노드의 IP 주소 제한

다음 시나리오에서 Astra Data Store 클러스터 생성이 실패합니다.

1. Kubernetes 작업자 노드의 IP 주소는 Astra Data Store에 할당되지 않은 인터페이스에서 호스팅됩니다.
2. IP 주소는 Astra Data Store 클러스터에 할당된 클러스터/데이터 IP 주소와 동일한 서브넷에 있습니다.

자세한 내용을 확인하십시오

- "알려진 제한 사항"

알려진 제한 사항

알려진 제한 사항은 제품의 이 조기 액세스 프로그램(EAP) 릴리스에서 지원하지 않거나 제품과 올바르게 상호 운용되지 않는 플랫폼, 장치 또는 기능을 식별합니다.

Astra Data Store 2022.05 조기 액세스 프로그램(EAP) 릴리스

이 릴리스에는 알려진 제한 사항이 포함되어 있습니다. 이러한 알려진 제한 사항을 주의 깊게 검토하십시오.

VMware 통합 제한 사항

다음 기능은 지원되지 않습니다.

- VM 워크플로:
 - 템플릿의 클론을 포함하여 VM을 클론 복제합니다
 - 이전에 VM 디스크를 제거한 후 VM 디스크를 연결합니다
- VVOL 복제
- VVOL 데이터 저장소를 삭제합니다
- 일등석 디스크(FCD)
- 마이그레이션 했습니다
- 연결 모드 vCenter 서버
- 규정 준수 경고 및 알림
- 다중 vCenter 및 다중 VASA Provider 구성
 - vCenter당 단일 VASA Provider 구성이 지원됩니다
- 스토리지 정책 기반 관리는 VM 레벨에서만 가능하며 VM 생성 후에는 수정할 수 없습니다.

여러 **VM** 디스크의 크기를 동시에 조정할 수 없습니다

이 릴리즈에서는 한 번에 하나의 디스크만 크기를 조정할 수 있습니다. 한 번에 여러 디스크의 크기를 조정하려고 하면 크기 조정 작업은 첫 번째 디스크에만 적용됩니다.

자세한 내용을 확인하십시오

- ["알려진 문제"](#)

개념

Astra Data Store에 대해 자세히 알아보십시오

Astra Data Store는 고객이 클라우드 네이티브 애플리케이션을 관리할 수 있도록 사내 데이터 센터를 위한 Kubernetes 네이티브 공유 파일 소프트웨어 정의 스토리지(SDS) 솔루션입니다. Astra Data Store는 NetApp 엔터프라이즈 데이터 관리와 함께 컨테이너 및 VM 워크로드 모두를 위한 네이티브 공유 파일 서비스를 제공합니다.

Astra Data Store를 사용하면 다음을 수행할 수 있습니다.

- * Kubernetes 컨테이너 워크로드 * 지원: 엔터프라이즈 데이터 관리 서비스 및 툴을 사용하는 데 적합합니다.
- * DevOps * 용 Kubernetes "서비스형 애플리케이션" 플랫폼 사용: 자동화되고 반복 가능한 서비스를 제공하고 개발자로부터 복잡성을 제거하는 탄력적인 소프트웨어 정의 셀프 서비스 플랫폼을 생성합니다.

Astra Data Store는 Astra 제품군의 일부입니다. 에 대해 자세히 알아보십시오 ["아스트라 가족"](#).

Astra Data Store 기능

Astra Data Store는 다음과 같은 기능을 통해 클라우드 네이티브 애플리케이션을 위한 Kubernetes 네이티브 엔드 스토리지 및 데이터 관리를 제공합니다.

- * Kubernetes 네이티브 공유 파일 서비스 *: 표준 NFS 클라이언트를 컨테이너 및 VM에 대한 통합 데이터 저장소로 사용하여 Kubernetes 네이티브 공유 파일 서비스를 제공합니다
- * 클라우드 확장 *: 동일한 리소스 풀에서 Kubernetes 네이티브 다중 병렬 파일 시스템을 제공하여 클라우드와 같은 확장 및 사용률을 달성하므로 스토리지와 클러스터를 따로 관리할 필요가 없습니다.
- * API 우선 접근 방식 *: 자동화된 클러스터 및 워크로드 관리에 필요한 코드로 인프라를 제공합니다.
- * 엔터프라이즈급 데이터 관리 *: 자동화된 애플리케이션 인식 데이터 보호 및 재해 복구 기능 제공:
 - * NetApp 기술 *: 스냅샷, 백업, 복제 및 클론 복제를 위해 NetApp 데이터 관리 기술을 활용하므로 사용자가 Kubernetes에서 엔터프라이즈 앱을 구축 및 구축할 수 있습니다.
 - * 복원력 *: Kubernetes 네이티브 워크로드에 복제 및 삭제 코딩 기술을 사용하여 뛰어난 복원력을 제공합니다.
 - * 데이터 효율성 *: 인라인 중복제거 및 압축 기능을 통해 확장 시 비용을 제어합니다.
- * 기존 환경에 적합 *: 마이크로서비스 기반 및 기존 워크로드 지원, 주요 Kubernetes 배포 서비스, 파일 스토리지 제공, 하드웨어 방식의 하드웨어인 Windows 담은 선택하여 실행할 수 있습니다.
- * NetApp Cloud Insights *와의 통합: 지속적인 낙관적인 견해를 관찰 가능성, 분석 및 모니터링합니다.

Astra Data Store를 시작하십시오

첫째, ["Astra Data Store 요구 사항에 대해 알아보십시오"](#).

그런 다음 ["시작하십시오"](#).

를 참조하십시오

- ["Astra 제품군 소개"](#)
- ["Astra Control Service 문서"](#)

- ["Astra Control Center 문서"](#)
- ["Astra Trident 문서"](#)
- ["Astra Control API를 사용합니다"](#)
- ["Cloud Insights 설명서"](#)
- ["ONTAP 설명서"](#)

Astra Data Store 배포 모델

Astra Data Store는 Kubernetes로 구축 및 오케스트레이션된 애플리케이션을 사용하여 호스트에서 직접 스토리지 드라이브를 관리합니다.

다음 옵션 중 하나를 사용하여 베어 메탈 또는 가상 서버에 Astra Data Store를 설치할 수 있습니다.

- 독립 실행형 전용 Kubernetes 클러스터에 구축하여 별도의 클러스터(독립 실행형 클러스터)에서 실행 중인 Kubernetes 애플리케이션에 영구 볼륨을 제공합니다.
- Kubernetes 클러스터에 구축할 경우 동일한 노드 풀(통합 클러스터)에서 다른 워크로드 애플리케이션을 호스팅할 수 있습니다.
- Kubernetes 클러스터에 구축할 경우 다른 노드 풀(disaggregated 클러스터)에서 다른 워크로드 애플리케이션을 호스팅할 수도 있습니다.

["Astra Data Store 하드웨어 요구 사항에 대해 자세히 알아보십시오"](#).

Astra Data Store는 Astra 제품군의 일부입니다. Astra 제품군 전체에 대한 자세한 내용은 을 참조하십시오 ["Astra 제품군 소개"](#).

Astra Data Store 에코시스템

Astra Data Store는 다음과 같이 작동합니다.

- * Astra Control Center *: 사내 환경에서 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 위해 Astra Control Center 소프트웨어를 사용합니다. Kubernetes 앱을 손쉽게 백업하고, 데이터를 다른 클러스터로 마이그레이션하고, 작동하는 애플리케이션 클론을 즉시 생성할 수 있습니다.

Astra Control Center는 ONTAP 또는 Astra Data Store 스토리지 백엔드가 포함된 Astra Trident 스토리지 백엔드를 통해 Kubernetes 클러스터를 지원합니다.

- * Astra Trident *: NetApp에서 관리하는 완전 지원되는 오픈 소스 스토리지 공급자 및 오케스트레이터로서, Astra Trident를 사용하면 Docker 및 Kubernetes에서 관리하는 컨테이너 형태의 애플리케이션용 스토리지 볼륨을 생성할 수 있습니다.

Astra Trident를 사용하여 Astra Data Store에 볼륨을 생성합니다.

- * Cloud Insights *: NetApp 클라우드 인프라 모니터링 툴인 Cloud Insights를 사용하면 Astra Control에서 관리하는 Kubernetes 클러스터의 성능과 활용률을 모니터링할 수 있습니다. Cloud Insights는 스토리지 사용량과 워크로드를 상호 연관시킵니다.

Astra Control에서 Cloud Insights 연결을 활성화하면 Astra Control UI 페이지에 원격 측정 정보가 표시됩니다. Cloud Insights는 Astra 데이터 저장소에서 관리되는 리소스에 대한 정보를 표시합니다.

Astra Data Store 인터페이스

다른 인터페이스를 사용하여 작업을 완료할 수 있습니다.

- * 웹 UI(사용자 인터페이스) *: Astra Control Service와 Astra Control Center는 앱을 관리, 마이그레이션 및 보호할 수 있는 동일한 웹 기반 UI를 사용합니다. 이 UI에는 Astra Data Store 볼륨에 대한 정보도 표시됩니다.
- * API *: Astra Control Service와 Astra Control Center는 동일한 Astra Control API를 사용합니다. API를 사용하면 UI를 사용할 때와 동일한 작업을 수행할 수 있습니다. Astra Control API를 사용하여 Astra Data Store에 대한 정보를 검색할 수도 있습니다.
- * kubectl Commands *: Astra Data Store와 함께 작업하려면 kubectl 명령을 직접 사용할 수 있습니다.
- * Kubernetes 확장 *: 또한 Astra Data Store Kubernetes API 확장을 사용할 수 있습니다.

사용자 정의 리소스 정의(CRD)는 Astra Data Store 운영자를 구축할 때 생성되는 Kubernetes REST API의 확장입니다. 외부 엔터티는 Kubernetes API 서버를 호출하여 CRD와 상호 작용합니다. Astra Data Store는 특정 CRD에 대한 업데이트를 확인한 다음 내부 REST API를 호출합니다.

를 참조하십시오

- ["Astra 제품군 소개"](#)
- ["Astra Control Service 문서"](#)
- ["Astra Control Center 문서"](#)
- ["Astra Trident 문서"](#)
- ["Astra Control API를 사용합니다"](#)
- ["Cloud Insights 설명서"](#)
- ["ONTAP 설명서"](#)

클러스터 확장

Astra Data Store는 클러스터의 여러 유형과 기능을 지원하는 노드입니다. 클러스터를 확장하는 경우 Astra Data Store는 성능 기능이 있는 노드를 추가할 수 있도록 지원하며, 이 경우 클러스터의 최소 성능 노드보다 성능이 낮은 노드만 추가하면 됩니다. 새 노드의 스토리지 용량은 기존 노드와 같아야 합니다. 확장 중에 새 노드를 비롯한 모든 노드는 에서 최소 요구사항을 충족해야 합니다 ["Astra 데이터 저장소 요구 사항"](#).

를 참조하십시오

- ["Astra 제품군 소개"](#)
- ["Astra Control Service 문서"](#)
- ["Astra Control Center 문서"](#)
- ["Astra Trident 문서"](#)
- ["Astra API를 사용합니다"](#)
- ["Cloud Insights 설명서"](#)
- ["ONTAP 설명서"](#)

Astra Data Store의 스토리지 효율성

Astra Data Store는 NetApp ONTAP 및 SolidFire 기술을 기반으로 다음과 같은 스토리지 효율성 기술을 사용합니다.

- * 씬 프로비저닝 *: 씬 프로비저닝된 볼륨은 스토리지가 미리 예약되지 않은 볼륨입니다. 대신 필요에 따라 스토리지가 동적으로 할당됩니다. 볼륨 또는 LUN의 데이터가 삭제되면 사용 가능한 공간이 스토리지 시스템으로 다시 해제됩니다.
- * 제로 블록 감지 및 제거 *: 씬 프로비저닝을 지원하는 Astra Data Store 스토리지 시스템은 0으로 제로화된 볼륨의 영역을 탐지하여 해당 공간을 재확보하고 다른 곳에서 사용할 수 있도록 합니다.
- * 압축 *: 압축은 압축 그룹에 데이터 블록을 결합하여 볼륨에 필요한 물리적 스토리지의 양을 줄이며, 각 블록이 단일 블록으로 저장됩니다. Astra Data Store는 전체 파일이 아닌 요청된 데이터가 들어 있는 압축 그룹만 압축 해제하므로 압축된 데이터를 기존 압축 방법보다 빠르게 읽을 수 있습니다.
- * 데이터 중복 제거 *: 중복 제거는 중복 블록을 버리고 단일 공유 블록에 대한 참조로 대체하여 볼륨(또는 AFF 애그리게이트의 모든 볼륨)에 필요한 스토리지 양을 줄입니다. 중복 제거된 데이터를 읽으면 일반적으로 성능 저하가 발생하지 않습니다. 쓰기 작업은 오버로드된 노드를 제외하고 무시할 만한 비용이 발생합니다.

이러한 모든 기능은 기본적으로 활성화되어 있습니다.

을 참조하십시오 ["스토리지 효율성 세부 정보"](#).

를 참조하십시오

- ["Astra 제품군 소개"](#)
- ["Astra Control Service 문서"](#)
- ["Astra Control Center 문서"](#)
- ["Astra Trident 문서"](#)
- ["Astra Control API를 사용합니다"](#)
- ["ONTAP 설명서"](#)

Astra Data Store의 보안

Astra Data Store는 여러 가지 방법을 사용하여 클라이언트 및 관리자의 스토리지 액세스 권한을 보호하고, 통신 및 데이터를 보호하며, 바이러스로부터 보호합니다.

Astra Data Store는 다음과 같은 보안 방법을 사용합니다.

- MTL(상호 전송 계층 보안)을 사용한 통신 암호화
- 기능에 대한 액세스를 제어하는 역할 기반 액세스 제어
- 배포 보안
- 인증서 관리
- 내부 및 외부 키 관리를 포함하여 유효 상태의 소프트웨어 암호화

를 참조하십시오

- ["Astra 제품군 소개"](#)
- ["Astra Control Service 문서"](#)
- ["Astra Control Center 문서"](#)
- ["Astra Trident 문서"](#)
- ["Astra Control API를 사용합니다"](#)
- ["Cloud Insights 설명서"](#)
- ["ONTAP 설명서"](#)

시작하십시오

Astra 데이터 저장소 요구 사항

먼저 귀사의 환경이 Astra Data Store 요구 사항을 충족하는지 확인하십시오.

Astra Data Store는 베어 메탈 및 VM 기반 구축을 모두 지원합니다. Astra Data Store 클러스터는 4개 이상의 작업자 노드가 있는 Kubernetes 클러스터에서 실행될 수 있습니다. Astra Data Store 소프트웨어는 동일한 Kubernetes 클러스터에서 실행 중인 다른 애플리케이션과 함께 사용할 수 있습니다.

- [\[Kubernetes worker node resources\]](#)
- [\[Hardware and software\]](#)
- [\[Networking\]](#)
- [\[Astra Trident\]](#)
- [\[Container Network Interfaces\]](#)
- [\[Licensing\]](#)



Astra Control Center에서 Astra Data Store 클러스터를 관리하려는 경우, Astra Data Store 클러스터가 를 충족하는지 확인하십시오 ["Astra Control Center에서 관리할 클러스터에 대한 요구 사항입니다"](#) 여기에 설명된 요구 사항 외에

Kubernetes 작업자 노드 리소스

Kubernetes 클러스터의 각 작업자 노드에 있는 Astra Data Store 소프트웨어에 할당하기 위해 필요한 리소스 요구사항은 다음과 같습니다.

리소스	최소	최대
Astra Data Store 클러스터에 있는 노드 수입니다	4	16
데이터 드라이브 수입니다	<ul style="list-style-type: none">• 3(별도의 캐시 디바이스가 있는 경우)• 4(캐시 디바이스가 없는 경우)	14
데이터 드라이브 크기입니다	100GiB	4TiB
선택적 캐시 디바이스의 수입니다	1(8GiB 크기)	해당 없음

Astra Data Store는 각 Kubernetes 작업자 노드에 대해 다음과 같은 vCPU 및 RAM 조합을 지원합니다.

- RAM 38GiB의 vCPU 9개
- RAM 94GiB의 vCPU 23개



최상의 쓰기 성능을 얻으려면 지연 시간이 짧은 전용 고성능 캐시 디바이스를 구성해야 합니다.

각 작업자 노드에는 다음과 같은 추가 요구 사항이 있습니다.

- Astra Data Store 로그 파일을 저장할 호스트 디스크(부팅)의 여유 공간 100GiB 이상
- 클러스터, 데이터 및 관리 트래픽을 위한 최소 1개의 10GbE 또는 더 빠른 네트워크 인터페이스 필요에 따라 추가 1GbE 또는 더 빠른 인터페이스를 사용하여 관리 트래픽을 분리할 수 있습니다.

하드웨어 및 소프트웨어

Astra Data Store 소프트웨어는 다음과 같은 하드웨어 플랫폼, 소프트웨어 및 스토리지 구성에서 검증되었습니다. 를 방문하십시오 ["NetApp 커뮤니티 지원"](#) Kubernetes 클러스터 구성이 다른 경우

하드웨어 플랫폼

- Dell R640
- Dell R740
- HPE DL360
- HPE DL380
- Lenovo SR630
- Lenovo SR650



VM 기반 배포에서는 에 호환되는 것으로 나열된 서버를 사용할 수도 있습니다 ["VMware 호환성 가이드"](#) 를 참조하십시오.

스토리지

Astra Data Store는 SATA, SAS 및 NVMe TLC SSD로 검증되었습니다.

VM 기반 배포의 경우 가상 디스크 또는 패스스루 드라이브로 제공된 드라이브를 사용할 수 있습니다.



- 호스트에서 하드웨어 RAID 컨트롤러 뒤에 SSD를 사용하는 경우 "통과" 모드를 사용하도록 하드웨어 RAID 컨트롤러를 구성합니다.
- 각 드라이브에는 고유한 일련 번호가 있어야 합니다. VM 생성 시 가상 머신 고급 설정에 `disk.enableuid=true` 속성을 추가합니다.

소프트웨어

- 하이퍼바이저: Astra Data Store는 vSphere 7.0을 통해 VMware 기반 VM 구축을 통해 검증되었습니다. KVM 기반 배포는 Astra Data Store에서 지원되지 않습니다.
- Astra Data Store는 다음과 같은 호스트 운영 체제에서 검증되었습니다.
 - Red Hat Enterprise Linux 8.4
 - Red Hat Enterprise Linux 8.2
 - Red Hat Enterprise Linux 7.9
 - Red Hat Enterprise Linux CoreOS(RHCOS)
 - CentOS 8
 - Ubuntu 20.04
- Astra Data Store는 다음과 같은 Kubernetes 배포에서 검증되었습니다.

- Red Hat OpenShift 4.6 - 4.9
- Google Anthos 1.8 - 1.10
- Kubernetes 1.20 ~ 1.23
- Rancher RKE 1.3.3



Astra Data Store에는 스토리지 프로비저닝 및 오케스트레이션을 위한 Astra Trident가 필요합니다. Astra Trident 버전 21.10.1 ~ 22.04가 지원됩니다. 를 참조하십시오 ["Astra Trident 설치 지침"](#).

네트워킹

Astra Data Store에는 MVIP를 위해 클러스터당 하나의 IP 주소가 필요합니다. MIP와 동일한 서브넷에서 사용되지 않거나 구성되지 않은 IP 주소여야 합니다. Astra Data Store 관리 인터페이스는 Kubernetes 노드의 관리 인터페이스와 동일해야 합니다.

또한 다음 표에 설명된 대로 각 노드를 구성할 수 있습니다.



이 표에는 MIP: 관리 IP 주소 CIP: 클러스터 IP 주소 MVIP: 관리 가상 IP 주소가 사용됩니다

구성	IP 주소가 필요합니다
노드당 1개의 네트워크 인터페이스	<ul style="list-style-type: none"> • 노드당 2개: <ul style="list-style-type: none"> ◦ MIP/CIP: 노드당 관리 인터페이스에서 사전 구성된 IP 주소 1개 ◦ 데이터 IP: MIP와 동일한 서브넷의 노드당 사용되지 않거나 구성되지 않은 IP 주소 1개
노드당 2개의 네트워크 인터페이스	<ul style="list-style-type: none"> • 노드당 3개: <ul style="list-style-type: none"> ◦ MIP: 노드당 관리 인터페이스에 사전 구성된 IP 주소 1개 ◦ CIP: MIP와 다른 서브넷의 노드별 데이터 인터페이스에 사전 구성된 IP 주소 1개 ◦ 데이터 IP: CIP와 동일한 서브넷에 있는 노드당 사용되지 않거나 구성되지 않은 IP 주소 1개



이러한 구성에는 VLAN 태그가 사용되지 않습니다.

방화벽 고려 사항

네트워크 방화벽 트래픽 필터링을 적용하는 환경에서는 아래 표에 나와 있는 포트 및 프로토콜에 들어오는 트래픽을 허용하도록 방화벽을 구성해야 합니다. IP 주소 열은 다음과 같은 약어를 사용합니다.

- MIP: 각 노드의 관리 인터페이스에 있는 기본 IP 주소입니다
- CIP: 각 노드의 클러스터 인터페이스에 있는 기본 IP 주소입니다
- DIP: 노드에 구성된 하나 이상의 데이터 IP 주소입니다

- MVIP: 하나의 클러스터 노드에 구성된 관리 가상 IP 주소입니다

포트/프로토콜	IP 주소입니다	목적	참고
111/TCP	딤	NFS 를 참조하십시오	데이터 IP는 런타임 시 클러스터 노드 간에 이동합니다. 각 노드는 모든 데이터 IP(또는 전체 서브넷)에 대해 이 트래픽을 허용해야 합니다.
442/TCP	MIP	API를 참조하십시오	
635/TCP	딤	NFS 를 참조하십시오	데이터 IP는 런타임 시 클러스터 노드 간에 이동합니다. 각 노드는 모든 데이터 IP(또는 전체 서브넷)에 대해 이 트래픽을 허용해야 합니다.
2010/UDP	CIP	클러스터/노드 검색	응답을 포함하여 UDP 포트 2010과 UDP 포트 2010을 오가는 유니캐스트 및 브로드캐스트 트래픽을 모두 포함합니다.
2049/TCP	딤	NFS 를 참조하십시오	데이터 IP는 런타임 시 클러스터 노드 간에 이동합니다. 각 노드는 모든 데이터 IP(또는 전체 서브넷)에 대해 이 트래픽을 허용해야 합니다.
2181-2183/TCP	CIP	분산 통신	
2443/TCP	MIP	API를 참조하십시오	
2443/TCP	MVIP	API를 참조하십시오	MVIP 주소는 모든 클러스터 노드에서 호스팅할 수 있으며 필요할 때 런타임에 재배포됩니다.
4000-4006/TCP	CIP	클러스터 내 RPC	
4045-4046/TCP	딤	NFS 를 참조하십시오	데이터 IP는 런타임 시 클러스터 노드 간에 이동합니다. 각 노드는 모든 데이터 IP(또는 전체 서브넷)에 대해 이 트래픽을 허용해야 합니다.
7700/TCP	CIP	세션 관리자	
9919/TCP	MIP	DMS API	
9920/TCP	딤	DMS REST 서버	

포트/프로토콜	IP 주소입니다	목적	참고
ICMP	CIP + DIP	노드 내 통신, 상태 점검	데이터 IP는 런타임 시 클러스터 노드 간에 이동합니다. 각 노드는 모든 데이터 IP(또는 전체 서브넷)에 대해 이 트래픽을 허용해야 합니다.

아스트라 트리덴트

Astra Data Store를 사용하려면 스토리지 프로비저닝 및 오케스트레이션을 위해 애플리케이션 Kubernetes 클러스터를 Astra Trident를 실행 중이어야 합니다. Astra Trident 버전 21.10.1 ~ 22.04가 지원됩니다. Astra Data Store는 로 구성할 수 있습니다 ["스토리지 백엔드"](#) Astra Trident를 사용하여 영구 볼륨 프로비저닝

컨테이너 네트워크 인터페이스

Astra Data Store는 다음 컨테이너 네트워크 인터페이스(CNIs)를 사용하여 검증되었습니다.

- RKE 클러스터용 Calico
- 바닐라 Kubernetes 클러스터용 Calico 및 Weave Net CNIs
- Red Hat OpenShift Container Platform(OCP)용 OpenShift SDN
- Google Anthos의 Cilium



- Cilium CNI와 함께 배포된 Astra Data Store에는 HostPort 지원을 위한 포트맵 플러그인이 필요합니다. 'cilium-config configMap'에 'cni-chaining-mode:portmap'을 추가하고 Cilium Pod를 다시 시작하여 CNI chaining 모드를 활성화할 수 있습니다.
- 방화벽 지원 구성은 Calico 및 OpenShift SDN CNIs에서만 지원됩니다.

라이센싱

Astra Data Store에는 전체 기능을 사용하려면 유효한 라이선스가 필요합니다.

["여기에서 등록하십시오"](#) Astra Data Store 라이선스를 취득합니다. 라이선스 다운로드 지침은 가입 후 발송됩니다.

다음 단계

를 봅니다 ["빠른 시작"](#) 개요.


를 참조하십시오

["Astra 데이터 저장소 제한"](#)

Astra Data Store를 빠르게 시작합니다


이 페이지에서는 Astra Data Store를 시작하는 데 필요한 단계에 대해 개괄적으로 설명합니다. 각 단계의 링크를 클릭하면 자세한 내용을 제공하는 페이지로 이동합니다.

사용해 보세요! Astra Data Store를 사용해 보고 싶은 경우 90일 평가판 라이선스를 사용할 수 있습니다. ["여기에서 등록하십시오"](#) Astra Data Store 라이선스를 취득하는 것입니다.

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png>
Kubernetes 클러스터 요구 사항을 검토합니다


- 클러스터가 정상 상태에서 4개 이상의 작업자 노드를 가지고 실행 중이어야 합니다.
- Astra Data Store 구축에 포함된 각 Kubernetes 작업자 노드에는 동일한 인터페이스 유형(SATA, SAS 또는 NVMe)의 SSD와 Astra Data Store 클러스터에 할당된 동일한 수의 드라이브가 있어야 합니다.
- 각 SSD에는 고유한 일련 번호가 있어야 합니다.

에 대해 자세히 알아보십시오 ["Astra 데이터 저장소 요구 사항"](#).

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png>
Astra Data Store를 다운로드하고 설치합니다

- 에서 Astra Data Store를 다운로드하십시오 ["NetApp Support 사이트"](#).
- 현지 환경에 Astra Data Store를 설치합니다.
- Astra Data Store 라이선스를 적용합니다.
- Astra Data Store 클러스터를 설치합니다.

에 대해 자세히 알아보십시오 ["Astra Data Store 설치 중"](#).

 <https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png>
일부 초기 설정 작업을 완료합니다

- Astra Trident를 설치합니다.
- Kubernetes 스냅샷 CRD(사용자 지정 리소스 정의) 및 컨트롤러를 설치합니다.
- Astra Data Store를 스토리지 백엔드로 설정합니다.
- 기본 Astra Data Store 스토리지 클래스를 생성합니다.
- 원격 측정 서비스를 위해 Astra Data Store를 구성합니다.

에 대해 자세히 알아보십시오 ["초기 설정 프로세스"](#).

Astra Data Store 설정을 마치면 다음 단계를 수행할 수 있습니다.

- 유지보수 모드에 노드 배치, 드라이브 교체 또는 노드 교체 등의 작업을 포함하여 클러스터를 관리하려면 `kubctl` 명령과 `kectl astrand` 확장을 사용하십시오. 에 대해 자세히 알아보십시오 ["Astra Data Store에서 kubelet 명령을 사용하는 방법"](#).
- 모니터링 엔드포인트를 구성합니다. 에 대해 자세히 알아보십시오 ["모니터링 엔드포인트 구성"](#).
- ["VMware와 함께 Astra Data Store를 사용하십시오"](#).

["Astra Data Store를 설치합니다"](#).

Astra Data Store를 설치합니다

Kubernetes 네이티브 명령을 사용하거나 Astra Control Center의 UI를 사용하여 Astra Data Store를 설치할 수 있습니다.

설치 옵션

- * Kubernetes 네이티브 명령 * 사용: Kubernetes 네이티브 명령을 사용하여 Astra Data Store를 설치하려면 [에](#) 설명된 설치 단계를 완료하십시오 [수행할 수 있습니다](#).
- * Astra Control Center * 포함: Astra Control Center를 사용하여 Astra Data Store를 설치하려면 [에](#) 설명된 설치 단계를 완료하십시오 [수행할 수 있습니다](#).

필요한 것

- "설치를 시작하기 전에 Astra Data Store 구축을 위한 환경을 준비하십시오".
- [에](#) 액세스합니다 "NetApp Support 사이트". "등록" NetApp Support 사이트 계정이 아직 없는 경우 평가판을 다운로드하십시오.
- A "NetApp 라이선스 파일(NLF)" Astra Data Store용. 라이선스 다운로드 지침이 사용자에게 전송됩니다 ["가입하세요"](#).
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubeconfig.
- [에](#) 대한 이해 "역할 및 권한" Astra Data Store에서 사용합니다.

Astra Data Store를 설치합니다

표준 Kubernetes 클러스터를 위한 Astra Data Store 설치 프로세스는 다음과 같은 고급 단계를 안내합니다. Red Hat OpenShift Container Platform(OCP) 환경의 추가 설치 단계도 설명되어 있습니다.

- [\[Download the Astra Data Store bundle and extract the images\]](#)
- [\[Copy the binary and push images to your local registry\]](#)
- [\[OpenShift procedure\]](#)
- [\[Install the Astra Data Store operator\]](#)
- [\[Deploy the Astra Data Store version YAML\]](#)
- [\[Apply the Astra Data Store license\]](#)
- [\[Install the Astra Data Store cluster\]](#)
- [\[Understand deployment-related events\]](#)



Google Anthos와 함께 Astra Data Store를 사용하려면 설치 단계를 완료하고 Anthos 환경에 Astra Data Store 클러스터를 추가하십시오.



Rancher RKE 환경과 함께 Astra Data Store를 설치하려면 설치 단계와 kubectl 명령을 위한 대체 rancher 명령을 수행합니다.

Astra Data Store 번들을 다운로드하고 이미지를 추출합니다

1. 에 로그인합니다 "[NetApp Support 사이트](#)" Astra Data Store 번들('Astra_Data_Store_2022.05.tar')을 다운로드하십시오.



이전 버전의 번들에 대한 지침은 을 참조하십시오 "[해당 버전에 대한 문서](#)".

2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify Astra_Data_Store_2022.05.pub -signature  
Astra_Data_Store_2022.05.sig 2022.12.01_ads.tar
```

3. 디렉토리 생성:

```
mkdir Astra_Data_Store_2022.05  
cd Astra_Data_Store_2022.05
```

4. 이미지 추출:

```
tar -xvf <path to tar file>/Astra_Data_Store_2022.05.tar
```



이미지는 작업 디렉토리 내에 생성된 'Astrads/images' 디렉토리에 압축이 풀립니다.

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

1. k8s kubectl 바이너리가 설치된 표준 경로로 이미지를 추출하는 데 사용한 디렉토리에서 kubectl-astrads 바이너리를 복사합니다("/usr/bin/"은 아래 예에서 경로로 사용됨). kubectl-astra 데이터 저장소 클러스터를 설치 및 관리하는 사용자 지정 kubectl 확장입니다.



kubbeck 명령을 사용하여 kubectl 바이너리가 설치된 경로를 찾습니다.

```
cp -p ./astrads/bin/kubectl-astrads /usr/bin/.
```

2. Astra Data Store 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.



아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

- a. 레지스트리에 로그인합니다.

```
docker login [your_registry_path]
```

- b. 환경 변수를 레지스트리 경로로 설정하여 Astra Data Store 이미지를 푸시합니다(예: REpo.company.com`).

```
export REGISTRY=repo.company.com/astrads
```

- c. 스크립트를 실행하여 이미지를 Docker에 로드하고, 이미지에 태그를 지정하고, 를 눌러 이미지를 로컬 레지스트리에 로드합니다.

```
for astraImageFile in $(ls astrads/images/*.tar) ; do
    astraImage=$(docker load --input ${astraImageFile} | sed 's~Loaded
image: ~~')
    astraImageShort=`echo $astraImage | sed 's~.*/~~'`
    docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
    docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR REGISTRY\]~'${REGISTRY}'~'
./astrads/manifests/*.yaml
```

OpenShift 절차

다음 절차는 Red Hat OpenShift Container Platform(OCP)에서의 배포에만 사용하십시오. OCP Kubernetes가 아닌 클러스터에 구축할 경우 이 절차를 건너뛰십시오.

- Namespace를 생성하여 Astra Data Store를 구축합니다
- 사용자 지정 SCC를 생성합니다
- 역할 및 역할 바인딩을 만듭니다

예 1. 세부 정보

모든 Astra Data Store 구성 요소를 설치할 네임스페이스 'astrads-system'을 만듭니다.

다음 단계는 Red Hat OpenShift Container Platform(OCP)에 배포용으로만 필요합니다.

1. 네임스페이스 만들기:

```
kubectl create -f ads_namespace.yaml
```

샘플: ads_namespace.YAML

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: operator
  name: astrads-system
```

OpenShift는 POD가 수행할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 사용합니다. 기본적으로 모든 컨테이너의 실행에는 제한된 SCC와 해당 SCC에 의해 정의된 기능만 부여됩니다.

제한된 SCC는 Astra Data Store 클러스터 포드에 필요한 권한을 제공하지 않습니다. 이 절차를 사용하여 Astra Data Store에 필요한 권한(샘플에 나열되어 있음)을 제공합니다.

사용자 지정 SCC를 Astra Data Store 네임스페이스의 기본 서비스 계정에 할당합니다.

다음 단계는 Red Hat OpenShift Container Platform(OCP)에 배포용으로만 필요합니다.

1. 사용자 지정 SCC 생성:

```
kubectl create -f ads_privileged_scc.yaml
```

샘플: ads_privileged_csC.yAML

```

allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
- '*'
allowedUnsafeSysctls:
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'ADS privileged. Grant with caution.'
    release.openshift.io/create-only: "true"
  name: ads-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups:
  type: RunAsAny
users:
- system:serviceaccount:astrads-system:default
volumes:
- '*'

```

2. OC Get SCC 명령을 사용하여 새로 추가한 SCC를 출력한다.

```
# oc get scc/ads-privileged
NAME          PRIV  CAPS  SELINUX  RUNASUSER  FSGROUP  SUPGROUP
PRIORITY  READONLYROOTFS  VOLUMES
ads-privileged true  ["*"] RunAsAny RunAsAny RunAsAny RunAsAny
<no value> false          ["*"]
#
```

Astra Data Store의 기본 서비스 계정에서 사용할 필수 역할 및 역할 바인딩을 만듭니다.

다음 YAML 정의는 Astra Data Store 리소스에 필요한 다양한 역할(rolebindings)을 'astrads.netapp.io' API 그룹'에 할당합니다.

다음 단계는 Red Hat OpenShift Container Platform(OCP)에 배포용으로만 필요합니다.

1. 정의된 역할 및 역할 바인딩을 생성합니다.

```
kubectl create -f oc_role_bindings.yaml
```

샘플: OC_ROLE_BINDINGS.YAML

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: privcrole
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - ads-privileged
  resources:
  - securitycontextconstraints
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-scc-rolebinding
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: privcrole
subjects:
```



```

- kind: ServiceAccount
  name: default
  namespace: astrads-system
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ownerref
  namespace: astrads-system
rules:
- apiGroups:
  - astrads.netapp.io
  resources:
  - '*/finalizers'
  verbs:
  - update
---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: or-rb
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ownerref
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system

```

개인 이미지 레지스트리를 구성합니다

특정 환경의 선택적 단계로서 암호를 사용하는 개인 레지스트리에서 이미지를 가져올 수 있도록 구성을 변경할 수 있습니다.

1. 이전 단계에서 이미 작성한 경우를 제외하고 'Astads-system' 네임스페이스를 만듭니다.

```
kubectl create namespace astrads-system
```

2. 비밀 만들기:

```
kubectl create secret docker-registry <secret-name> -n astrads-system
--docker-server=<registry name> --docker-username= <registry username>
--docker-password=<registry user password>
```

3. 서비스 계정에 비밀 구성 정보 추가:

```
kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name":
"<secret-name>"}]}' -n astrads-system
```



이러한 변경 사항은 사용자가 적용할 때 적용됩니다 [Astra Data Store 운영자를 설치합니다](#).

Astra Data Store 운영자를 설치합니다

1. Astra Data Store 매니페스트 나열:

```
ls astrads/manifests/*.yaml
```

응답:

```
astrads/manifests/monitoring_operator.yaml
astrads/manifests/astradscluster.yaml
astrads/manifests/astradsversion.yaml
astrads/manifests/astradsoperator.yaml
astrads/manifests/vasa_asup_certs.yaml
astrads/manifests/manifest.yaml
astrads/manifests/configuration.yaml
```

2. kubectk apply를 사용하여 운영자를 배치한다.

```
kubectl apply -f ./astrads/manifests/astradsoperator.yaml
```

응답:



네임스페이스 응답은 표준 설치를 수행했는지 또는 을 수행했는지에 따라 다를 수 있습니다 ["OpenShift Container Platform 설치"](#).

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsadddrives.astrads.n
etapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
```

```
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscLOUDsnapshots.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscLusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradskeyproviders.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslICENSES.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodemANagements.ast
rads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssearkeyrotatereques
ts.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsversions.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.netapp.io created
role.rbac.authorization.k8s.io/astrads-astrads-system-admin-role created
role.rbac.authorization.k8s.io/astrads-astrads-system-reader-role
created
role.rbac.authorization.k8s.io/astrads-astrads-system-writer-role
created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
role.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-admin-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-reader-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-writer-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-editor-
role created
```

```
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-  
editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-  
viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-  
editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-  
viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-editor-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-  
editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-  
viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsqospolicy-viewer-  
role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-editor-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-viewer-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role  
created  
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumeeditor-  
role created
```

```

clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumeviewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
rolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
secret/astrads-autosupport-certs created
secret/astrads-webhook-server-cert created
service/astrads-webhook-service created
deployment.apps/astrads-operator created

```

3. Astra Data Store 운영자 POD가 시작되고 실행 중인지 확인합니다.

```
kubectl get pods -n astrads-system
```

응답:

NAME	READY	STATUS	RESTARTS	AGE
astrads-operator-5ffb94fbf-7ln4h	1/1	Running	0	17m

Astra Data Store 버전 YAML을 배포하십시오

1. kubectl apply를 이용한 구축:

```
kubectl apply -f ./astrads/manifests/astradsversion.yaml
```

2. Pod가 실행 중인지 확인합니다.

```
kubectl get pods -n astrads-system
```

응답:

NAME	READY	STATUS	RESTARTS
AGE			
astrads-cluster-controller-7f6f884645-xxf2n	1/1	Running	0
117s			
astrads-ds-nodeinfo-2jqnk	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-dbk7v	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-rn9tt	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-vsmhv	1/1	Running	0
2m7s			
astrads-license-controller-fb8fd56bc-bxq7j	1/1	Running	0
2m2s			
astrads-operator-5ffb94fbf-7ln4h	1/1	Running	0
2m10s			

Astra Data Store 라이선스를 적용합니다

1. NetApp에서 구입한 NetApp 라이선스 파일(NLF)을 적용합니다. 명령을 실행하기 전에 현재 클러스터 이름('<Astra-Data-Store-cluster-name>')을 입력합니다 [배포로 이동합니다](#) 또는 이미 배포되어 있고 사용권 파일('<file_path/file.txt>')에 대한 경로가 있습니다.

```
kubectl astrads license add --license-file-path <file_path/file.txt> --ads-cluster-name <Astra-Data-Store-cluster-name> -n astrads-system
```

2. 라이선스가 추가되었는지 확인합니다.

```
kubectl astrads license list
```

응답:

NAME	ADSCUSTER	VALID	PRODUCT
EVALUATION	ENDDATE	VALIDATED	
e100000006-ads-capacity	astrads-example-cluster	true	Astra Data Store
true	2023-01-23	2022-04-04T14:38:54Z	

Astra Data Store 클러스터를 설치합니다

1. YAML 파일을 엽니다.

```
vim ./astrads/manifests/astradscluster.yaml
```

2. YAML 파일에서 다음 값을 편집합니다.



YAML 파일의 간단한 예는 다음과 같습니다.

- (필수) * 메타데이터 *: metadata에서 이름 문자열을 클러스터 이름으로 변경합니다. 이 이름은 사용 시 사용한 클러스터 이름과 같아야 합니다 [라이선스를 적용합니다](#).
- (필수) * Spec *: 'sepec'에서 다음 필수 값을 변경합니다.
 - "adsNodeConfig" 값을 라이선스 및 Astra Data Store 설치 크기에 따라 설치에 필요한 값으로 변경합니다.
 - 소형: 9 CPU 및 38 메모리
 - 보통: 23 CPU 및 94 메모리
 - (선택 사항) 'adsNodeSelector' 섹션 주위의 코멘트를 제거합니다. 선택한 작업자 노드 풀에만 설치하도록 Astra Data Store를 제한하려면 이 옵션을 구성합니다.
 - (선택 사항) Astra Data Store 클러스터에서 사용해야 하는 4-16개의 특정 노드 수를 지정합니다.
 - 클러스터의 작업자 노드에서 라우팅할 수 있는 부동 관리 IP의 IP 주소로 mVIP 문자열을 변경합니다.
 - adsDataNetworks에서 NetApp 볼륨을 마운트할 호스트에서 라우팅할 수 있는 침표로 구분된 부동 IP 주소 목록("주소")을 추가합니다. 노드당 하나의 부동 IP 주소를 사용합니다. Astra Data Store 노드만큼 데이터 네트워크 IP 주소가 적어도 몇 개 있어야 합니다. Astra Data Store의 경우 4개 이상의 주소를 의미하며, 나중에 클러스터를 확장할 계획이라면 최대 16개까지 가능합니다.
 - adsDataNetworks에서 데이터 네트워크에서 사용하는 넷마스크를 지정한다.
 - adsNetworkInterfaces에서 '<mgmt_interface_name>' 및 '<cluster_and_storage_interface_name>' 값을 관리, 클러스터 및 스토리지에 사용할 네트워크 인터페이스 이름으로 바꿉니다. 이름을 지정하지 않으면 노드의 기본 인터페이스가 관리, 클러스터 및 스토리지 네트워킹에 사용됩니다. 또한 adsNetworkInterfaces 섹션에 대한 주석도 제거해야 합니다.



클러스터 및 스토리지 네트워크는 동일한 인터페이스에 있어야 합니다. Astra Data Store 관리 인터페이스는 Kubernetes 노드의 관리 인터페이스와 동일해야 합니다.

- (선택 사항) * monitoringConfig *: 를 구성하려는 경우 [운전자 모니터링](#) (모니터링을 위해 Astra Control

Center를 사용하지 않는 경우 선택 사항) 섹션에서 메모를 제거하고 에이전트 CR(모니터링 운영자 리소스)이 적용되는 네임스페이스(기본값은 NetApp 모니터링)를 추가한 다음 이전 단계에서 사용한 레지스트리('your_registry_path')의 경로를 추가합니다.

- d. (선택 사항) * autoSupportConfig *: 를 유지합니다 "AutoSupport" 프록시를 구성할 필요가 없는 경우 기본값:
- proxyURL의 경우 AutoSupport 번들 전송에 사용할 포트를 사용하여 프록시 URL을 설정합니다.



간결성을 위해 아래의 YAML 샘플에서 몇 가지 의견이 제거되었습니다.

```
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 38
    # [Optional] Specify node selector labels to select the nodes for
    # creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
        # Specify the network interface names to use for management, cluster
        # and storage networks.
        # If none are specified, the node's primary interface will be used for
        # management, cluster and storage networking.
        # To move the cluster and storage networks to a different interface
        # than management, specify all three interfaces to use here.
        # NOTE: The cluster and storage networks need to be on the same
        # interface.
  adsNetworkInterfaces:
    managementInterface: "<mgmt_interface_name>"
    clusterInterface: "<cluster_and_storage_interface_name>"
    storageInterface: "<cluster_and_storage_interface_name>"
    # [Optional] Provide a monitoring config to be used to setup/configure
    # a monitoring agent.
  # monitoringConfig:
  #   namespace: "netapp-monitoring"
  #   repo: "[YOUR_REGISTRY]"
  autoSupportConfig:
```



```

autoUpload: true
enabled: true
coreDumpUpload: false
historyRetentionCount: 25
destinationURL: "https://support.netapp.com/put/AsupPut"
# ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
# proxyURL:
periodic:
- schedule: "0 0 * * *"
  periodicconfig:
  - component:
      name: storage
      event: dailyMonitoring
      userMessage: Daily Monitoring Storage AutoSupport bundle
      nodes: all
  - component:
      name: controlplane
      event: daily
      userMessage: Daily Control Plane AutoSupport bundle

```

3. "kubctl apply"를 사용하여 클러스터를 구축합니다.

```
kubect1 apply -f ./astrads/manifests/astradscluster.yaml
```

4. 클러스터 생성 작업이 완료될 때까지 몇 분 정도 기다린 후 Pod가 실행 중인지 확인합니다.

```
kubect1 get pods -n astrads-system
```

샘플 반응:

NAME	READY	STATUS	
RESTARTS AGE			
astrads-cluster-controller-7c67cc7f7b-2jww2 7h31m	1/1	Running	0
astrads-deployment-support-788b859c65-2qjkn 12d	3/3	Running	19
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 5d1h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-pwk42 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 8h	1/1	Running	0
astrads-ds-nodeinfo-gcmj8 12d	1/1	Running	1
astrads-ds-nodeinfo-j826x 12d	1/1	Running	3
astrads-ds-nodeinfo-vdthh 12d	1/1	Running	3
astrads-ds-nodeinfo-xwgsf 12d	1/1	Running	0
astrads-ds-support-828vw 5d2h	2/2	Running	2
astrads-ds-support-astrads-example-cluster-cfzts 8h	2/2	Running	0
astrads-ds-support-astrads-example-cluster-nzkkr 7h49m	2/2	Running	15
astrads-ds-support-astrads-example-cluster-xxbnp 5d2h	2/2	Running	1
astrads-license-controller-86c69f76bb-s6fb7 8h	1/1	Running	0
astrads-operator-79ff8fbb6d-vpz9m 8h	1/1	Running	0

5. 클러스터 배포 진행 상태 확인:

```
kubectl get astradscluster -n astrads-system
```

샘플 반응:

NAME AGE	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster 10.x.x.x 13m	created	2022.05.0-X	e100000006	

배포 관련 이벤트를 이해합니다

클러스터 배치 중에는 작동 상태가 공간에서 진행 중 상태로 변경되어야 합니다. 클러스터 구축은 약 8~10분간 지속됩니다. 구축하는 동안 클러스터 이벤트를 모니터링하려면 다음 명령 중 하나를 실행합니다.

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

다음은 배포 중에 발생하는 주요 이벤트입니다.

이벤트	메시지와 중요성
ControlPlaneNotesSelected 를 선택합니다	ADS 클러스터에 연결할 [number] 컨트롤 플레인 노드를 성공적으로 선택했습니다. Astra Data Store 운영자는 CPU, 메모리, 스토리지 및 네트워킹으로 Astra Data Store 클러스터를 생성할 수 있는 충분한 노드를 식별했습니다.
ADSClusterCreateInProgress(ADSClusterCreateInProgress)	Astra Data Store 클러스터 컨트롤러가 클러스터 생성 작업을 시작했습니다.
ADSClusterCreateSuccess	클러스터가 생성되었습니다.

클러스터의 상태가 "In progress(진행 중)"로 변경되지 않는 경우 운영자 로그에서 노드 선택에 대한 자세한 내용을 확인하십시오.

```
kubectl logs -n astrads-system <astrads operator pod name>
```

클러스터의 상태가 "In progress(진행 중)"로 고착된 경우 클러스터 컨트롤러의 로그를 확인하십시오.

```
kubectl logs -n astrads-system <astrads cluster controller pod name>
```

Astra Control Center를 사용하여 Astra Data Store를 설치합니다

Astra Control Center와 함께 Astra Data Store를 배포하고 사용하려면 다음을 수행하십시오.

무엇을 '필요로 할거야

- 검토했습니다 [일반 Astra Data Store 필수 구성 요소입니다](#).
- Astra Control Center를 설치했습니다.

단계

1. ["Astra Control Center를 사용하여 Astra Data Store를 구축합니다"](#).

다음 단계

- * Kubernetes 네이티브 구축 및 타사 배포 *: 추가 작업을 수행하여 Astra Data Store 구축을 완료합니다 ["설정 작업"](#).
- * Astra Control Center *: Astra Control Center를 사용하여 Astra Data Store를 구축한 경우 이러한 사항을 따를 필요가 없습니다 ["설정 작업"](#) 추가 모니터링 옵션을 구성하지 않는 경우. Astra Data Store를 구축한 후 Astra Control Center UI를 사용하여 다음과 같은 작업을 수행할 수 있습니다.
 - ["Astra Data Store 자산의 상태를 모니터링합니다"](#).
 - ["Astra Data Store 백엔드 스토리지를 관리합니다"](#).
 - ["노드, 디스크 및 지속적인 PVC\(Volume Claim\) 모니터링"](#).

Astra Data Store 구성 요소를 설정합니다

먼저 해 ["독립 실행형 Astra Data Store를 설치했습니다"](#) 몇 가지 문제를 해결했습니다 ["환경 전제 조건"](#) Astra Trident를 설치하고, Kubernetes 스냅샷 기능을 구성하고, 스토리지 백엔드를 설정하고, 기본 스토리지 클래스를 생성합니다.



Astra Control Center를 사용하여 Astra Data Store를 배포하는 경우, 설정을 계획하지 않는 한 다음 단계를 따를 필요가 없습니다 [추가 모니터링 옵션](#).

- [\[Install Astra Trident\]](#)
- [\[Install Kubernetes snapshot CRDs and Controller\]](#)
- [\[Set up Astra Data Store as storage backend\]](#)
- [\[Create a default Astra Data Store storage class\]](#)
- [\[Configure Astra Data Store monitoring\]](#)

Astra Trident를 설치합니다

Astra Data Store의 경우 Astra Trident 21.10.1 이상을 설치해야 합니다. 다음 옵션 중 하나를 사용하여 Astra Trident를 설치할 수 있습니다.

- ["tridentctl을 사용하여 Astra Trident를 설치합니다"](#).
- ["Trident 연산자를 사용하여 Astra Trident를 설치합니다"](#).



Trident 연산자는 수동으로 또는 Hrom을 사용하여 배포할 수 있습니다.

Kubernetes 스냅샷 CRD 및 컨트롤러를 설치합니다

영구 볼륨 청구(PVC) 스냅샷을 생성하려면 Kubernetes 스냅샷 CRD 및 컨트롤러가 필요합니다. 사용자 환경에 CRD 및 컨트롤러가 아직 설치되어 있지 않은 경우 다음 명령을 실행하여 설치하십시오.



다음 명령 예제에서는 디렉터리를 '/trident'로 가정합니다. 그러나 사용하는 디렉터리는 YAML 파일을 다운로드하는 데 사용한 디렉터리일 수 있습니다.

무엇을 '필요로 할거야

- "설치를 시작하기 전에 Astra Data Store 구축을 위한 환경을 준비하십시오".
- 를 다운로드합니다 "Kubernetes 스냅샷 컨트롤러 YAML 파일":
 - 설정 - 스냅샷 - 컨트롤러.YAML
 - RBAC-스냅샷-컨트롤러.YAML
- 를 다운로드합니다 "YAML CRD":
 - snapshot.storage.k8s.io_volumesnapshotClasses.YAML
 - snapshot.storage.k8s.io_volumesnapshot내용물.YAML
 - snapshot.storage.k8s.io_volumesnapshots.YAML

단계

1. snapshot.storage.k8s.io_volumesnapshotclasses.YAML:

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
```

응답:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotclasses.snapshot.storage.k8s.io configured
```

2. snapshot.storage.k8s.io_volumesnapshot내용물 적용.YAML:

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
```

응답:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotcontents.snapshot.storage.k8s.io configured
```

3. snapshot.storage.k8s.io_volumesnapshots를 적용합니다.YAML:

```
kubectl apply -f trident/snapshot.storage.k8s.io_volumesnapshots.yaml
```

응답:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshots.snapshot.s  
storage.k8s.io configured
```

4. 설정 적용 - 스냅샷 - 컨트롤러.YAML:

```
kubectl apply -f trident/setup-snapshot-controller.yaml
```

응답:

```
deployment.apps/snapshot-controller configured
```

5. RBAC-스냅샷-컨트롤러 적용.YAML:

```
kubectl apply -f trident/rbac-snapshot-controller.yaml
```

응답:

```
serviceaccount/snapshot-controller configured  
clusterrole.rbac.authorization.k8s.io/snapshot-controller-runner  
configured  
clusterrolebinding.rbac.authorization.k8s.io/snapshot-controller-role  
configured  
role.rbac.authorization.k8s.io/snapshot-controller-leaderelection  
configured  
rolebinding.rbac.authorization.k8s.io/snapshot-controller-leaderelection  
configured
```

6. CRD YAML 파일이 적용되었는지 확인합니다.

```
kubectl get crd | grep volumesnapshot
```

샘플 반응:

```
astradsvolumesnapshots.astrads.netapp.io      2021-08-04T17:48:21Z
volumesnapshotclasses.snapshot.storage.k8s.io 2021-08-04T22:05:49Z
volumesnapshotcontents.snapshot.storage.k8s.io 2021-08-04T22:05:59Z
volumesnapshots.snapshot.storage.k8s.io       2021-08-04T22:06:17Z
```

7. 스냅샷 컨트롤러 파일이 적용되었는지 확인합니다.

```
kubectl get pods -n kube-system | grep snapshot
```

샘플 반응:

```
snapshot-controller-7f58886ff4-cdh78
1/1      Running    0          13s
snapshot-controller-7f58886ff4-tmrd9
1/1      Running    0          32s
```

Astra Data Store를 스토리지 백엔드로 설정합니다

ads_backend.json 파일에 스토리지 백엔드 매개 변수를 구성하고 Astra Data Store 스토리지 백엔드를 생성합니다.

단계

1. 보안 터미널을 사용하여 ads_backend.json을 생성한다.

```
vi ads_backend.json
```

2. JSON 파일 구성:



JSON 샘플은 다음 단계를 따릅니다.

- ""클러스터"" 값을 Astra Data Store 클러스터의 클러스터 이름으로 변경합니다.
- ""namespace"" 값을 볼륨 생성에 사용할 네임스페이스로 변경합니다.
- 이 백엔드에 대한 내보내기 정책 CR을 대신 설정하지 않는 한 ""autoExportPolicy"" 값을 "true"로 변경합니다.
- 액세스를 허용할 IP 주소로 ""autoExportCIDR"" 목록을 채웁니다. 모두 허용하려면 0.0.0.0/0을 사용하십시오.
- ""kubecoreconfig"" 값을 보려면 다음을 수행합니다.
 - 공백 없이 .kubecore/config YAML 파일을 JSON 형식으로 변환 및 최소화:

변환 예:

```
python3 -c 'import sys, yaml, json;
json.dump(yaml.load(sys.stdin), sys.stdout, indent=None)' <
~/.kube/config > kubeconf.json
```

ii. base64로 인코딩하고 base64 출력을 " kubecononfig " 값에 사용합니다.

인코딩 예:

```
cat kubeconf.json | base64 | tr -d '\n'
```



```

{
  "version": 1,
  "storageDriverName": "astrads-nas",
  "storagePrefix": "",
  "cluster": "example-1234584",
  "namespace": "astrads-system",
  "autoExportPolicy": true,
  "autoExportCIDRs": ["0.0.0.0/0"],
  "kubeconfig": "<base64_output_of_kubeconf_json>",
  "debugTraceFlags": {"method": true, "api": true},
  "labels": {"cloud": "on-prem", "creator": "trident-dev"},
  "defaults": {
    "qosPolicy": "silver"
  },
  "storage": [
    {
      "labels": {
        "performance": "extreme"
      },
      "defaults": {
        "qosPolicy": "gold"
      }
    },
    {
      "labels": {
        "performance": "premium"
      },
      "defaults": {
        "qosPolicy": "silver"
      }
    },
    {
      "labels": {
        "performance": "standard"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    }
  ]
}

```

3. Trident 설치 프로그램을 다운로드한 디렉토리로 이동합니다.

```
cd <trident-installer or path to folder containing tridentctl>
```

4. 스토리지 백엔드를 생성합니다.

```
./tridentctl create backend -f ads_backend.json -n trident
```

샘플 반응:

```
+-----+-----+
+-----+-----+-----+
|      NAME      | STORAGE DRIVER |                      UUID
| STATE  | VOLUMES |
+-----+-----+
+-----+-----+-----+
| example-1234584 | astrads-nas    | 2125fa7a-730e-43c8-873b-
6012fcc3b527 | online |          0 |
+-----+-----+
+-----+-----+-----+
```

기본 **Astra Data Store** 스토리지 클래스를 생성합니다

Astra Trident 기본 스토리지 클래스를 생성하고 스토리지 백엔드에 적용합니다.

단계

1. 트리덴트 CSI 스토리지 클래스를 생성합니다.

a. ADS_SC_Example.YAML 생성:

```
vi ads_sc_example.yaml
```

예:

```

allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2022-05-09T18:05:21Z"
  name: ads-silver
  resourceVersion: "3361772"
  uid: 1o023456-da4b-51e3-b430-3aa1e3bg111a
mountOptions:
- vers=4
parameters:
  backendType: astrads-nas
  selector: performance=premium
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```

b. 트리덴트 CSI 생성:

```
kubectl create -f ads_sc_example.yaml
```

응답:

```
storageclass.storage.k8s.io/trident-csi created
```

2. 스토리지 클래스가 추가되었는지 확인합니다.

```
kubectl get storageclass
```

응답:

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION	AGE		
ads-silver	csi.trident.netapp.io	Delete	Immediate
true	6h29m		

3. Trident 설치 프로그램을 다운로드한 디렉토리로 이동합니다.

```
cd <trident-installer or path to folder containing tridentctl>
```

4. Astra Trident 백엔드가 기본 스토리지 클래스 매개 변수로 업데이트되었는지 확인합니다.

```
./tridentctl get backend -n trident -o yaml
```

샘플 반응:

```
items:
- backendUUID: 2125fa7a-730e-43c8-873b-6012fcc3b527
  config:
    autoExportCIDRs:
    - 0.0.0.0/0
    autoExportPolicy: true
    backendName: ""
    cluster: example-1234584
    credentials: null
    debug: false
    debugTraceFlags:
      api: true
      method: true
    defaults:
      exportPolicy: default
      qosPolicy: bronze
      size: 1G
      snapshotDir: "false"
      snapshotPolicy: none
    disableDelete: false
    kubeconfig: <ID>
    labels:
      cloud: on-prem
      creator: trident-dev
    limitVolumeSize: ""
    namespace: astrads-system
    nfsMountOptions: ""
    region: ""
    serialNumbers: null
    storage:
    - defaults:
        exportPolicy: ""
        qosPolicy: gold
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
      region: ""
```

```

supportedTopologies: null
zone: ""
- defaults:
  exportPolicy: ""
  qosPolicy: silver
  size: ""
  snapshotDir: ""
  snapshotPolicy: ""
labels:
  performance: premium
region: ""
supportedTopologies: null
zone: ""
- defaults:
  exportPolicy: ""
  qosPolicy: bronze
  size: ""
  snapshotDir: ""
  snapshotPolicy: ""
labels:
  performance: standard
region: ""
supportedTopologies: null
zone: ""
storageDriverName: astrads-nas
storagePrefix: ""
supportedTopologies: null
version: 1
zone: ""
configRef: ""
name: example-1234584
online: true
protocol: file
state: online
storage:
  example-1234584_pool_0:
    name: example-1234584_pool_0
    storageAttributes:
      backendType:
        offer:
          - astrads-nas
      clones:
        offer: true
      encryption:
        offer: false
    labels:

```

```

    offer:
      cloud: on-prem
      creator: trident-dev
      performance: extreme
  snapshots:
    offer: true
storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_1:
  name: example-1234584_pool_1
  storageAttributes:
    backendType:
      offer:
        - astrads-nas
    clones:
      offer: true
    encryption:
      offer: false
    labels:
      offer:
        cloud: on-prem
        creator: trident-dev
        performance: premium
    snapshots:
      offer: true
storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_2:
  name: example-1234584_pool_2
  storageAttributes:
    backendType:
      offer:
        - astrads-nas
    clones:
      offer: true
    encryption:
      offer: false
    labels:
      offer:
        cloud: on-prem
        creator: trident-dev
        performance: standard
    snapshots:
      offer: true

```

```
storageClasses:
  - ads-silver
supportedTopologies: null
volumes: []
```

Astra Data Store 모니터링을 구성합니다

(선택 사항) 다른 원격 측정 서비스를 통해 모니터링하도록 Astra Data Store를 구성할 수 있습니다. Astra Data Store 모니터링을 위해 Astra Control Center를 사용하지 않거나 모니터링을 추가 엔드포인트로 확장하려면 이 절차를 권장합니다.

Astra Data Store 인스턴스가 독립 실행형 배포이거나 Cloud Insights를 사용하여 원격 측정을 모니터링하거나 Elastic과 같은 타사 엔드포인트로 로그를 스트리밍하는 경우 모니터링 연산자를 설치할 수 있습니다.



Astra Control Center 배포의 경우 모니터링 운영자가 자동으로 구성됩니다. 다음 절차의 처음 두 명령을 건너뛸 수 있습니다.

모니터링을 설정하기 전에 "Astra-system" 네임스페이스에 활성 Astra 데이터 저장소 클러스터가 필요합니다.

단계

1. 다음 설치 명령을 실행합니다.

```
kubectl apply -f ./manifests/monitoring_operator.yaml
```

2. 모니터링을 위해 Astra Data Store 구성:

```
kubectl astrads monitoring -n netapp-monitoring -r [YOUR REGISTRY] setup
```

3. EMS 로그를 Elastic 끝점으로 스트리밍하도록 Astra Data Store 구성:

```
kubectl astrads monitoring es --port <portname> --host <hostname>
```

Astra Data Store 조기 액세스 프로그램(EAP) 릴리스 제한

Astra Data Store에는 조기 액세스 프로그램 동안 다음과 같은 리소스 제한이 있습니다.

리소스	최소	최대
Astra Data Store 클러스터에 있는 노드 수입니다	4	16
노드당 영구 볼륨 수입니다	해당 없음	50
볼륨 크기	20MiB	2TiB

리소스	최소	최대
볼륨당 스냅샷	0	1023
볼륨당 클론	0	9
노드당 VM	0	50

Astra Data Store에 대한 질문과 대답

Astra Data Store Early Access Program 릴리스의 설치, 구성, 업그레이드 및 문제 해결에 대해 자주 묻는 질문에 대한 답변을 찾아보십시오.

일반적인 질문

- Astra Data Store Early Access Program 릴리스를 프로덕션에 사용할 수 있습니까? * 아니요 Astra Data Store는 엔터프라이즈급 복원력을 제공하도록 설계 및 개발되었지만 운영 워크로드를 대상으로 하는 Astra Data Store의 초기 액세스 프로그램 릴리스는 아닙니다.
- 가상 머신 워크로드에 Astra Data Store를 사용할 수 있습니까? * 예. Astra Data Store는 Kubernetes 및 VMware VVOL 워크로드를 모두 지원합니다.

을 참조하십시오 ["VMware를 사용한 Astra Data Store에 대해 자세히 알아보십시오"](#).

- VMware vSphere를 사용하여 Astra Data Store를 관리할 수 있습니까? * 예. Astra Data Store는 VMware vSphere용 NetApp Astra Plugin을 사용하여 vCenter에서 기본적으로 관리할 수 있습니다. 을 참조하십시오 ["VMware 설치의 Astra Data Store 구성 요소를 관리합니다"](#).
- Astra Data Store는 해당 기능을 위해 다른 NetApp 제품에 의존할 수 있습니까? *

예. Astra Data Store를 사용하려면 NetApp CSI 드라이버 Astra Trident 버전 21.10.1 이상이 워크로드 Kubernetes 클러스터에 배포되어야 합니다. 에 대해 자세히 알아보십시오 ["Astra 데이터 저장소 요구 사항"](#).

VMware 워크플로우와 VMware vSphere용 NetApp Astra Plugin을 사용하려면 Astra Control Center가 필요합니다.

Astra Data Store 클러스터를 스토리지 백엔드로 사용하는 애플리케이션은 를 사용할 수 있습니다 ["Astra 제어 센터"](#) 데이터 보호, 재해 복구, Kubernetes 워크로드 마이그레이션을 비롯한 애플리케이션 인식 데이터 관리 기능을 활용할 수 있습니다.

- Astra Data Store 클러스터를 어떻게 관리할 수 있습니까? * kubectl 명령을 사용하거나 Kubernetes API 확장을 사용하여 Astra Data Store 리소스를 관리할 수 있습니다.

kubbeck astrads 명령어에는 편의에 따라 사용 및 플래그 문서를 제공하는 '-h' 스위치가 포함되어 있습니다.

- Astra 데이터 저장소 클러스터 메트릭을 어떻게 모니터링할 수 있습니까? * Cloud Insights를 사용하여 Astra 데이터 저장소 메트릭을 모니터링할 수 있습니다(참조) ["Cloud Insights를 통해 메트릭을 모니터링합니다"](#)) 또는 Kubernetes 네이티브 모니터링(참조 ["Prometheus 및 Grafana로 모니터링합니다"](#))를 클릭합니다.

로그를 모니터링할 수도 있습니다. 을 참조하십시오 ["이벤트 로그를 구성하고 모니터링합니다"](#).

- Kubernetes 클러스터의 ONTAP 또는 다른 스토리지 공급자와 함께 Astra Data Store를 사용할 수 있습니까? * 예. Astra Data Store를 애플리케이션 클러스터의 다른 스토리지 공급자와 함께 사용할 수 있습니다.
- Astra Data Store에서 Kubernetes 클러스터를 제거할 경우 Astra Trident를 제거할 수 있습니까? * Astra Data

Store를 제거하면 클러스터에서 Astra Trident를 제거할 수 없습니다. Astra Trident를 제거해야 하는 경우 별도로 제거해야 합니다.

라이센싱

- Astra Data Store에 라이선스가 필요합니까? * 예, Astra Data Store에 조기 액세스 프로그램을 위한 평가 NetApp 라이선스 파일(NLF)이 필요합니다.

을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#).

- Astra Data Store 평가판 라이선스는 얼마나 유효합니까? * Astra Data Store 라이선스의 기본 기간은 다운로드 날짜로부터 90일입니다.

Kubernetes 클러스터에 Astra Data Store 설치 및 사용

- 베어 메탈 또는 가상 머신에서 실행 중인 Kubernetes 클러스터에 Astra Data Store를 설치할 수 있습니까? * 예. Astra Data Store는 베어 메탈 또는 vSphere VM에서 실행되는 Kubernetes 클러스터에 설치할 수 있습니다. 을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#).
- Astra Data Store용 Kubernetes의 지원되는 버전은 무엇입니까? *

Astra Data Store는 v1.20 이상과 호환되는 Kubernetes 배포판과 연동됩니다. 그러나 현재 일부 Kubernetes 배포에서 검증되지 않았습니다. 에 대해 자세히 알아보십시오 ["Astra 데이터 저장소 요구 사항"](#).

- My Kubernetes 클러스터가 4개 이상의 작업자 노드입니다. Astra 데이터 저장소를 설치할 수 있습니까? * 예. Astra Data Store 클러스터는 Kubernetes 클러스터의 4개 작업자 노드에 먼저 구축해야 합니다. 구축한 후 최대 16개의 작업자 노드까지 클러스터를 확장할 수 있습니다.
- Astra Data Store는 개인 레지스트리의 오프라인 설치를 지원합니까? * 예. Astra Data Store는 로컬 레지스트리에서 오프라인으로 설치할 수 있습니다. 을 참조하십시오 ["Astra Data Store를 설치합니다"](#).
- Astra Data Store를 사용하려면 인터넷 연결이 필요합니까? * 아니요, Astra Data Store Early Access Program에는 인터넷 연결이 필요하지 않습니다. 하지만 원격 측정 번들을 주기적으로 전송하려면 NetApp AutoSupport 백엔드에 연결하는 것이 좋습니다. 이러한 연결은 직접 또는 프록시를 통해 가능합니다.
- Astra Data Store에서 사용하는 역할과 권한은 무엇입니까? * Astra Data Store 운영자를 구축하려면 kudo 관리자여야 합니다.

Astra Data Store에는 노드 선택에 사용되는 노드 리소스를 검색하기 위한 "astrads-ds-nodeinfo"라는 특별 데모가 있습니다.

또한 운영자는 권한이 있는 Kubernetes 작업을 사용하여 선택한 작업자 노드에 스토리지 클러스터의 컨테이너를 설치하여 Astra Data Store 스토리지 클러스터를 구축합니다.

- Astra Data Store 설치를 위해 업데이트해야 하는 매니페스트 파일은 무엇입니까? * 에서 다운로드한 Astra Data Store 번들의 목록을 통해 확인하십시오 ["NetApp Support 사이트"](#), 다음과 같은 매니페스트가 제공됩니다.
- Astasscluster.YAML
- Astassoperator.YAML
- Astradsversion.YAML을 참조하십시오
- monitoring_operator.YAML

배포별 구성으로 "astradsccluster.yaml" 매니페스트를 업데이트해야 합니다. 을 참조하십시오 ["Astra Data Store를 설치합니다"](#).

문제 해결 및 지원

Astra Data Store를 사용하면 NetApp Containers Slack 채널을 사용하여 커뮤니티 지원에 액세스할 수 있습니다. 이 채널은 NetApp Support 및 기술 마케팅 엔지니어가 모니터링합니다.

"NetApp 컨테이너 여유 채널"

을 참조하십시오 ["Astra Data Store 지원 작업"](#).

- 지원 케이스를 제출하거나 빠른 질문에 대한 명확한 설명을 요청하는 방법은 무엇입니까? * 지원 케이스를 제출하거나 빠른 질문에 대한 설명을 받으려면 에 대해 문제나 질문을 신고하십시오 ["NetApp 컨테이너 여유 채널"](#). NetApp Support는 최선의 노력을 통해 최선의 지원을 제공합니다.
- 새 기능에 대한 요청을 어떻게 제기합니까? * 지원되는 구성 또는 기능에 대한 질문이 있는 경우 astra.feedback@netapp.com 으로 문의하십시오.
- 지원 로그 번들을 생성하려면 어떻게 합니까? * 를 참조하십시오 ["지원 번들을 생성합니다"](#) Astra Data Store용 지원 로그 번들을 설정 및 다운로드하는 방법에 대한 지침은
- Astra Data Store에서 내 Kubernetes 노드를 찾을 수 없습니다. 이 문제를 해결하려면 어떻게 합니까? * 를 참조하십시오 ["Astra Data Store를 설치합니다"](#).
- IPv6 주소를 관리, 데이터 및 클러스터 네트워크에 사용할 수 있습니까? * 아니요. Astra Data Store는 IPv4 주소만 지원합니다. IPv6 지원은 Astra Data Store의 향후 릴리스에 추가될 예정입니다.
- Astra Data Store에서 볼륨을 프로비저닝할 때 어떤 NFS 버전을 사용합니까? * Astra Data Store는 Kubernetes 애플리케이션에 프로비저닝된 모든 볼륨에 대해 NFS v4.1을 지원하고 VMware 워크로드용으로 프로비저닝된 모든 볼륨에 대해 NFSv3을 지원합니다.

을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#) 및 ["Astra 데이터 저장소 제한"](#).

Astra Data Store 업그레이드 중

- Astra Data Store Preview 릴리스에서 업그레이드할 수 있습니까? * 예. Astra Data Store Early Access Program 릴리즈에서 향후 릴리즈로 업그레이드할 수 있습니다.

Astra Data Store를 사용하십시오

kubeck 명령을 사용하여 Astra Data Store 리소스를 관리합니다

kubtl 명령을 사용하거나 Kubernetes API 확장을 사용하여 Astra Data Store 리소스를 관리할 수 있습니다.

샘플 앱을 배포하는 방법에 대한 자세한 내용은 을 참조하십시오 ["테스트 응용 프로그램을 배포합니다"](#).

클러스터 유지 보수 정보는 를 참조하십시오 ["클러스터를 관리합니다"](#).

무엇을 '필요로 할거야

- 에 설치한 Astra Data Store kubeck 플러그인 ["Astra Data Store를 설치합니다"](#)

Astra Data Store용 Kubernetes 사용자 지정 API 리소스를 나열합니다

Kubernetes 내의 kubtl 명령을 사용하여 Astra Data Store 클러스터와 상호 작용하고 상태를 관찰할 수 있습니다.

'api-resources' 명령에 나열된 각 항목은 Astra Data Store에서 클러스터 관리를 위해 내부적으로 사용하는 Kubernetes 사용자 정의 리소스 정의(CRD)를 나타냅니다.

이 목록은 나중에 보여진 것처럼 입력을 줄이기 위해 각 Astra Data Store 개체의 짧은 이름을 얻는 데 특히 유용합니다.

1. Astra Data Store용 Kubernetes 사용자 API 리소스 목록을 표시합니다.

```
kubect1 api-resources --api-group astrads.netapp.io
```

응답:

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
astradsautosupports	adsas	astrads.netapp.io/v1alpha1	true	
AstraDSAutoSupport				
astradscLOUDsnapshots	adscs	astrads.netapp.io/v1alpha1	true	
AstraDSCloudSnapshot				
astradsclusters	adscl	astrads.netapp.io/v1alpha1	true	
AstraDSCluster				
astradsexportpolicies	adsep	astrads.netapp.io/v1alpha1	true	
AstraDSEExportPolicy				
astradsfaileddrives	adsfd	astrads.netapp.io/v1alpha1	true	
AstraDSFailedDrive				
astradsllicenses	adsli	astrads.netapp.io/v1alpha1	true	
AstraDSLICENSE				
astradsnfsoptions	adsnf	astrads.netapp.io/v1alpha1	true	
AstraDSNfsOption				
astradsnodeinfoes	adsni	astrads.netapp.io/v1alpha1	true	
AstraDSNodeInfo				
astradsnodemanagements	adsnm	astrads.netapp.io/v1alpha1	true	
AstraDSNodeManagement				
astradsqospolicies	adsqp	astrads.netapp.io/v1alpha1	true	
AstraDSQosPolicy				
astradsversions	adsve	astrads.netapp.io/v1alpha1	true	
AstraDSVersion				
astradsvolumeFiles	adsvf	astrads.netapp.io/v1alpha1	true	
AstraDSVolumeFiles				
astradsvolumes	adsvo	astrads.netapp.io/v1alpha1	true	
AstraDSVolume				
astradsvolumesnapshots	adsvs	astrads.netapp.io/v1alpha1	true	
AstraDSVolumeSnapshot				

2. Kubernetes 클러스터의 모든 현재 Astra Data Store 객체를 가져오려면 "kubect get ads-a" 명령을 사용하십시오.

```
kubect1 get ads -A
```

응답:

NAMESPACE	NAME	AGE
astrads-system	astradsqospolicy.astrads.netapp.io/bronze	45h
astrads-system	astradsqospolicy.astrads.netapp.io/gold	45h
astrads-system	astradsqospolicy.astrads.netapp.io/silver	45h

NAMESPACE	NAME			
STATUS	VERSION	SERIAL NUMBER	MVIP	AGE

```
astrads-system    astradscluster.astrads.netapp.io/astrads-cluster-9f1
created    arda-9.11.1    e000000009    10.224.8.146    46h
```

```

NAMESPACE      NAME
AGE
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
```

```

NAMESPACE      NAME      AGE
astrads-system  astradsversion.astrads.netapp.io/astradsversion  46h
```

```

NAMESPACE      NAME      AGE
astrads-system  astradsvolumefiles.astrads.netapp.io/test23      27h
astrads-system  astradsvolumefiles.astrads.netapp.io/test234     27h
astrads-system  astradsvolumefiles.astrads.netapp.io/test2345    4h22m
```

```

NAMESPACE      NAME      SIZE    IP
CLUSTER      CREATED
astrads-system  astradsvolume.astrads.netapp.io/test234    21Gi
172.25.123.123  astrads-cluster-9f1  true
astrads-system  astradsvolume.astrads.netapp.io/test2345    21Gi
172.25.123.123  astrads-cluster-9f1  true
```

```

NAMESPACE      NAME
SEQUENCE COMPONENT      EVENT      TRIGGER      PRIORITY      SIZE
STATE
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
adsclustercreatesuccess-20211214t 9      controlplane
adsclustercreatesuccess k8sEvent  notice  0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
daily-20211215t0      15      controlplane  daily
periodic  notice  0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
daily-20211216t0      20      controlplane  daily
periodic  notice  0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/storage-
callhome.dbs.cluster.cannot.sync.blocks 10      storage
callhome.dbs.cluster.cannot.sync.blocks  firetapEvent  emergency  0
uploaded
```

NAMESPACE	NAME	ADSCluster
VALID PRODUCT	EVALUATION ENDDATE	VALIDATED
astrads-system	astradsllicense.astrads.netapp.io/e0	astrads-cluster-
9f1 true	Astra Data Store true	2022-02-07 2021-12-16T20:43:23Z

3. 간단한 이름 중 하나를 사용하여 클러스터에 있는 볼륨의 현재 상태를 표시합니다.

```
kubectl get adsvo -A
```

응답:

NAMESPACE	NAME	SIZE	IP	CLUSTER
CREATED				
astrads-system	test234	21Gi	172.25.138.109	astrads-cluster-
9f1c99f true				
astrads-system	test2345	21Gi	172.25.138.111	astrads-cluster-
9f1c99f true				

kubeck 확장명에 대한 도움말 옵션을 사용합니다

NetApp은 라이선스 추가, 노드 관리, 문제 해결, Astra Data Store 클러스터 확장 등의 Astra Data Store 클러스터 관리 작업을 수행할 수 있도록 kubectl 명령에 대한 "astrs"를 확장하였습니다. 이 확장에는 내선사용 방법과 작업에 대한 정보를 제공하는 '-h' 옵션이 포함되어 있습니다.

1. Astra Data Store의 kubectl 확장명에 대한 모든 명령어 도움말을 보여줌:

```
kubectl astrads -h
```

응답:

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

Available Commands:
  asup          Manage AutoSupport
  clusters      Manage clusters
  drives        Manage drives in a cluster
  faileddrive   Manage drive replacement in a cluster
  help          Help about any command
  license       Manage license in the astrads cluster
```

maintenance	Manage maintenance status of a node
monitoring	Manage Monitoring Output
nodes	Manage nodes in a cluster

Flags:

<code>--as</code> string operation	Username to impersonate for the operation
<code>--as-group</code> stringArray operation, this flag can be groups.	Group to impersonate for the operation, this flag can be repeated to specify multiple groups.
<code>--cache-dir</code> string cache")	Default HTTP cache directory (default "/u/arda/.kube/http-cache")
<code>--certificate-authority</code> string certificate authority	Path to a cert file for the certificate authority
<code>--client-certificate</code> string for TLS	Path to a client certificate file for TLS
<code>--client-key</code> string	Path to a client key file for TLS
<code>--cluster</code> string cluster to use	The name of the kubeconfig cluster to use
<code>--context</code> string context to use	The name of the kubeconfig context to use
<code>-h, --help</code>	help for astrads
<code>--insecure-skip-tls-verify</code> certificate will not be checked your HTTPS connections insecure	If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure
<code>--kubeconfig</code> string use for CLI requests.	Path to the kubeconfig file to use for CLI requests.
<code>-n, --namespace</code> string for this CLI request	If present, the namespace scope for this CLI request
<code>--request-timeout</code> string before giving up on a single should contain a 1s, 2m, 3h). timeout requests.	The length of time to wait for a server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests. (default "0")
<code>-s, --server</code> string Kubernetes API server	The address and port of the Kubernetes API server
<code>--token</code> string to the API server	Bearer token for authentication to the API server
<code>--user</code> string	The name of the kubeconfig user

to use

2. 명령에 대한 자세한 내용은 'astrads[command]--help'를 사용하십시오.

```
kubectl astrads asup collect --help
```

응답:

Collect the autosupport bundle by specifying the component to collect.
It will default to manual event.

Usage:

```
astrads asup collect [flags]
```

Examples:

```
# Control plane collection
```

```
kubectl astrads collect --component controlplane example1
```

```
# Storage collection for single node
```

```
kubectl astrads collect --component storage --nodes node1 example2
```

```
# Storage collection for all nodes
```

```
kubectl astrads collect --component storage --nodes all example3
```

```
# Collect but don't upload to support
```

```
kubectl astrads collect --component controlplane --local example4
```

NOTE:

```
--component storage and --nodes <name> are mutually inclusive.
```

```
--component controlplane and --nodes <name> are mutually  
exclusive.
```

Flags:

```
-c, --component string      Specify the component to collect:  
[storage , controlplane , vasaprovider, all]
```

```
-d, --duration int          Duration is the duration in hours from  
the startTime for collection  
of AutoSupport.
```

```
-e, --event string          Specify the callhome event to trigger.  
(default "manual")
```

```
-f, --forceUpload           Configure an AutoSupport to upload if  
it is in the compressed state  
and not
```


the 'local' option or if	uploading because it was created with
disabled	automatic uploads of AutoSupports is
	at the cluster level.
-h, --help	help for collect
-l, --local	Only collect and compress the
autosupport bundle. Do not upload	to support.
	Use 'download' to copy the collected
bundle after it is in	the 'compressed' state
	Specify nodes to collect for storage
--nodes string	
component. (default "all")	
-t, --startTime string	StartTime is the starting time for
collection of AutoSupport.	
	This should be in the ISO 8601 date
time format.	
	Example format accepted:
	2021-01-01T15:20:25Z, 2021-01-
01T15:20:25-05:00	
-u, --usermessage string	UserMessage is the additional message
to include in the	
	AutoSupport subject.
	(default "Manual event trigger from
CLI")	

테스트 응용 프로그램을 배포합니다

다음은 Astra Data Store에서 사용할 수 있는 테스트 애플리케이션을 구축하는 단계입니다.

이 예에서는 Hrom 리포지토리를 사용하여 Bitnami의 MongoDB 차트를 배포합니다.

무엇을 '필요로 할거야

- Astra Data Store 클러스터가 구축 및 구성되었습니다
- Trident 설치가 완료되었습니다

단계

1. Bitnami에서 Helm repo 추가:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. MongoDB 구축:

```
helm install mongohelm4 --set persistence.storageClass=trident-csi
bitnami/mongodb --namespace=ns-mongodb --create-namespace
```

3. MongoDB Pod의 상태를 확인합니다.

```
~% kubectl get pods -n ns-mongodb
```

NAME	READY	STATUS	RESTARTS	AGE
mongodb-9846ff8b7-rfr4r	1/1	Running	0	67s

4. MongoDB에서 사용되는 영구 볼륨 클레임(PVC)을 확인합니다.

```
~% kubectl get pvc -n ns-mongodb
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
mongodb	Bound	pvc-1133453a-e2f5-48a5	8Gi	RWO	trident-csi	97s

5. kubelet 명령 'Get astradsvolume'을 사용하여 볼륨을 나열합니다.

```
~% kubectl get astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-system
```

NAME	SIZE	IP	CLUSTER	CREATED
pvc-1133453a-e2f5-48a5	8830116Ki	10.192.2.192	jai-ads	true

6. kubect describe astradsvolume을 사용하여 볼륨을 설명합니다.

```
~% kubectl describe astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-
system
```

Name: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07

Namespace: astrads-system

Labels: astrads.netapp.io/cluster=jai-ads
astrads.netapp.io/mip=10.192.1.39
astrads.netapp.io/volumeUUID=cf33fd38-a451-596c-b656-61b8270d2b5e
trident.netapp.io/cloud=on-prem
trident.netapp.io/creator=trident-dev
trident.netapp.io/performance=premium

Annotations: provisioning: {"provisioning":{"cloud":"on-prem","creator":"trident-dev","performance":"premium"}}
trident: {"trident":{"version":"21.10.0-test.jenkins-trident-stable-v21.10-

```

2+e03219ce37294d9ba54ec476bbe788c1a7772548","backendUUID":"","platform":
...
API Version:  astrads.netapp.io/v1alpha1
Kind:          AstraDSVolume
Metadata:
  Creation Timestamp:  2021-12-08T19:35:26Z
  Finalizers:
    trident.netapp.io/astradsvolume-finalizer
    astrads.netapp.io/astradsvolume-finalizer
  Generation:  1
  Managed Fields:
    API Version:  astrads.netapp.io/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          f:astrads.netapp.io/cluster:
          f:astrads.netapp.io/mip:
          f:astrads.netapp.io/volumeUUID:
        f:status:
          .:
          f:cluster:
          f:conditions:
          f:created:
          f:displayName:
          f:exportAddress:
          f:internalName:
          f:mip:
          f:permissions:
          f:qosPolicy:
          f:requestedSize:
          f:restoreCacheSize:
          f:size:
          f:snapshotReservePercent:
          f:state:
          f:volumePath:
          f:volumeUUID:
    Manager:      cluster-controller
    Operation:    Update
    Time:         2021-12-08T19:35:32Z
    API Version:  astrads.netapp.io/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        f:exportPolicy:
    Manager:      dms-controller

```

```

Operation:      Update
Subresource:    status
Time:           2021-12-08T19:35:32Z
API Version:    astrads.netapp.io/v1alpha1
Fields Type:    FieldsV1
fieldsV1:
  f:metadata:
    f:annotations:
      .:
    f:provisioning:
    f:trident:
  f:finalizers:
    v:"trident.netapp.io/astradsvolume-finalizer":
  f:labels:
    .:
    f:trident.netapp.io/cloud:
    f:trident.netapp.io/creator:
    f:trident.netapp.io/performance:
  f:spec:
    .:
    f:cluster:
    f:displayName:
    f:exportPolicy:
    f:noSnapDir:
    f:permissions:
    f:qosPolicy:
    f:size:
    f:snapshotReservePercent:
    f:type:
    f:volumePath:
Manager:        trident_orchestrator
Operation:      Update
Time:           2021-12-08T19:35:34Z
Resource Version: 12007115
UID:            d522ae4f-e793-49ed-bbe0-9112d7f9167b
Spec:
  Cluster:      jai-ads
  Display Name:  pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  Export Policy: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  No Snap Dir:  true
  Permissions:  0777
  Qos Policy:   silver
  Size:         9042036412
  Snapshot Reserve Percent: 5
  Type:         ReadWrite
  Volume Path:  /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07

```

```

Status:
Cluster:  jai-ads
Conditions:
  Last Transition Time:    2021-12-08T19:35:32Z
  Message:                Volume is online
  Reason:                 VolumeOnline
  Status:                 True
  Type:                   AstraDSVolumeOnline
  Last Transition Time:    2021-12-08T19:35:32Z
  Message:                Volume creation request was successful
  Reason:                 VolumeCreated
  Status:                 True
  Type:                   AstraDSVolumeCreated
Created:                  true
Display Name:             pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Export Address:           10.192.2.192
Export Policy:            pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Internal Name:            pvc_1133453a_e2f5_48a5_a06c_d14b8aa7be07
Mip:                      10.192.1.192
Permissions:              777
Qos Policy:               silver
Requested Size:            9042036412
Restore Cache Size:       0
Size:                     8830116Ki
Snapshot Reserve Percent: 5
State:                    online
Volume Path:              /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Volume UUID:              cf33fd38-a451-596c-b656-61b8270d2b5e
Events:
  Type      Reason          Age    From                      Message
  ----      -
  Normal    VolumeCreated    3m9s   ADSClusterController     Volume creation
request was successful

```

Astra Data Store 클러스터를 관리합니다

Astra Data Store에서 `kubctl` 명령을 사용하여 클러스터를 관리할 수 있습니다.

- [\[Add a node\]](#)
- [\[Remove a node\]](#)
- [\[Place a node in maintenance mode\]](#)
- [\[Add drives to a node\]](#)
- [\[Replace a drive\]](#)

무엇을 '필요로 할거야

- kubectl 및 kubectl-astrads 플러그인이 설치된 시스템. 을 참조하십시오 ["Astra Data Store를 설치합니다"](#).

노드를 추가합니다

추가하려는 노드는 Kubernetes 클러스터의 일부여야 하며 클러스터의 다른 노드와 비슷한 구성을 가져야 합니다.



Astra Control Center를 사용하여 노드를 추가하려면 을 참조하십시오 ["스토리지 백엔드 클러스터에 노드를 추가합니다"](#).

단계

1. 새 노드의 데이터 IP가 Astra Data Store 클러스터 CR에 아직 포함되어 있지 않은 경우 다음을 수행합니다.
 - a. 클러스터 CR을 편집하고 "adsDataNetworks" * Addresses * 필드에 추가 데이터 IP를 추가합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl edit astradscluster CLUSTER_NAME -n astrads-system
```

응답:

```
adsDataNetworks:
  -addresses: dataIP1,dataIP2,dataIP3,dataIP4,NEW_DATA_IP
```

- a. CR을 저장합니다.
- b. Astra Data Store 클러스터에 노드를 추가합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

2. 그렇지 않으면 노드를 추가하면 됩니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

3. 노드가 추가되었는지 확인합니다.

```
kubectl astrads nodes list
```

노드를 제거합니다

Astra Data Store와 함께 kubectl 명령을 사용하여 클러스터의 노드를 제거합니다.

단계

1. 모든 노드 나열:

```
kubectl astrads nodes list
```

응답:

NODE NAME	NODE STATUS	CLUSTER NAME
sti-rx2540-534d...	Added	cluster-multinodes-21209
sti-rx2540-535d...	Added	cluster-multinodes-21209
...		

2. 제거할 노드를 표시합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes remove NODE_NAME
```

응답:

```
Removal label set on node sti-rx2540-534d.lab.org  
Successfully updated ADS cluster cluster-multinodes-21209 desired node  
count from 4 to 3
```

노드를 제거하도록 표시한 후 노드 상태가 활성에서 현재로 바뀌어야 합니다.

3. 제거된 노드의 'Present' 상태를 확인한다.

```
kubectl get nodes --show-labels
```

응답:

NAME	STATUS	ROLES
sti-astramaster-050.lab.org v1.20.0 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-astramaster-050.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=	Ready	control-plane,master 3h39m
sti-rx2540-556a.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-556b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-534d.lab.org v1.20.0 astrads.netapp.io/storage-cluster-status=present,astrads.netapp.io/storage-node-removal=,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m
sti-rx2540-557b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true	Ready	worker 3h38m

4. 노드에서 Astra Data Store를 제거합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes uninstall NODE_NAME
```

5. 클러스터에서 노드가 제거되었는지 확인합니다.

```
kubectl astrads nodes list
```

노드가 Astra Data Store에서 제거됩니다.

노드를 유지보수 모드로 전환합니다

호스트 유지보수 또는 패키지 업그레이드를 수행해야 하는 경우 노드를 유지보수 모드로 전환해야 합니다.



노드는 이미 Astra Data Store 클러스터에 포함되어 있어야 합니다.

노드가 유지보수 모드일 때는 클러스터에 노드를 추가할 수 없습니다. 이 예제에서는 노드 "nhcitj1525"를 유지보수 모드로 설정합니다.

단계

1. 노드 세부 정보를 표시합니다.

```
kubectl get nodes
```

응답:

NAME	STATUS	ROLES	AGE	VERSION
nhcitjj1525	Ready	<none>	3d18h	v1.20.0
nhcitjj1526	Ready	<none>	3d18h	v1.20.0
nhcitjj1527	Ready	<none>	3d18h	v1.20.0
nhcitjj1528	Ready	<none>	3d18h	v1.20.0
scs000039783-1	Ready	control-plane,master	3d18h	v1.20.0

2. 노드가 유지보수 모드가 아닌지 확인합니다.

```
kubectl astrads maintenance list
```

응답(유지보수 모드에 있는 노드가 이미 없음):

NAME	NODE NAME	IN MAINTENANCE	MAINTENANCE STATE	MAINTENANCE VARIANT
------	-----------	----------------	-------------------	---------------------

3. 유지보수 모드를 활성화합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads maintenance create CR_NAME --node-name=NODE_NAME  
--variant=Node
```

예를 들면 다음과 같습니다.

```
kubectl astrads maintenance create maint1 --node-name="nhcitjj1525"
--variant=Node
```

응답:

```
Maintenance mode astrads-system/maint1 created
```

4. 노드 나열:

```
kubectl astrads nodes list
```

응답:

NODE NAME	NODE STATUS	CLUSTER NAME
nhcitjj1525	Added	ftap-astra-012
...		

5. 유지보수 모드의 상태를 점검합니다.

```
kubectl astrads maintenance list
```

응답:

NAME	NODE NAME	IN MAINTENANCE	MAINTENANCE STATE
MAINTENANCE VARIANT			
node4	nhcitjj1525	true	ReadyForMaintenance Node

유지보수 모드는 거짓으로 시작해 참으로 바뀝니다. 유지 보수 상태가 PreparingForMaintenance에서 ReadyforMaintenance로 바뀝니다.

6. 노드 유지보수가 완료된 후 유지보수 모드를 비활성화합니다.

```
kubectl astrads maintenance update maint1 --node-name="nhcitjj1525"
--variant=None
```

7. 노드가 더 이상 유지보수 모드가 아닌지 확인합니다.

```
kubectl astrads maintenance list
```

노드에 드라이브를 추가합니다

Astra Data Store와 함께 kubectl 명령을 사용하여 Astra Data Store 클러스터의 노드에 물리적 또는 가상 드라이브를 추가합니다.

무엇을 '필요로 할거야

- 다음 기준을 충족하는 하나 이상의 드라이브:
 - 노드에 이미 설치되었거나(물리적 드라이브) 노드 VM(가상 드라이브)에 추가되었습니다.
 - 드라이브에 파티션이 없습니다
 - 드라이브가 현재 클러스터에서 사용되고 있지 않습니다
 - 드라이브 물리적 용량이 클러스터의 라이선스 물리적 용량을 초과하지 않음(예: CPU 코어당 2TB의 스토리지를 라이선스를 부여하면 10개 노드로 구성된 클러스터에 최대 20TB의 물리적 드라이브 용량이 있음)
 - 드라이브가 노드에 있는 다른 활성 드라이브의 크기입니다



Astra Data Store에는 노드당 드라이브를 16개 이상 필요로 하지 않습니다. 17번째 드라이브를 추가하려고 하면 드라이브 추가 요청이 거부됩니다.

단계

1. 클러스터 설명:

```
kubectl astrads clusters list
```

응답:

CLUSTER NAME	CLUSTER STATUS	NODE COUNT
cluster-multinodes-21209	created	4

2. 클러스터 이름을 기록합니다.

3. 클러스터의 모든 노드에 추가할 수 있는 드라이브를 표시합니다. cluster_name을 클러스터 이름으로 바꿉니다.

```
kubectl astrads drives adddrive show-available --cluster=CLUSTER_NAME
```

응답:

```

Node: node1.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c290e16d52479a9af5eac c290e16d52479a9af5eac 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c2935798df68355dee0be c2935798df68355dee0be 0
100 GiB false

Node: node2.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.

Node: node3.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c29ee82992ed7a36fc942 c29ee82992ed7a36fc942 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c29312aa362469fb3da9c c29312aa362469fb3da9c 0
100 GiB false

Node: node4.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.

```

4. 다음 중 하나를 수행합니다.

- 사용 가능한 모든 드라이브의 이름이 동일한 경우 해당 노드에 동시에 추가할 수 있습니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads drives adddrive create --cluster=CLUSTER_NAME --name
REQUEST_NAME --drivesbyname all=DRIVE_NAME
```

- 드라이브의 이름이 다른 경우 각 노드에 하나씩 추가할 수 있습니다(추가해야 하는 각 드라이브에 대해 이 단계를 반복해야 함). 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads drives adddrive create --cluster=CLUSTER_NAME --name
REQUEST_NAME --drivesbyname NODE_NAME=DRIVE_NAME
```

Astra Data Store가 드라이브 추가 요청을 생성하고 요청 결과와 함께 메시지가 나타납니다.

드라이브를 교체합니다

클러스터에서 드라이브가 고장난 경우 데이터 무결성을 보장하기 위해 가능한 한 빨리 드라이브를 교체해야 합니다. 드라이브에 장애가 발생하면 클러스터 CR 노드 상태, 클러스터 상태 정보 및 메트릭 끝점에서 장애가 발생한 드라이브에 대한 정보를 볼 수 있습니다. 다음 예제 명령을 사용하여 오류가 발생한 드라이브 정보를 볼 수 있습니다.

노드 **Statuses.driveStatuses**에서 장애가 발생한 드라이브를 표시하는 클러스터의 예

```
kubectl get adsc1 -A -o yaml
```

응답:

```
...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
...
nodeStatuses:
  - driveStatuses:
    - driveID: 31205e51-f592-59e3-b6ec-185fd25888fa
      driveName: scsi-36000c290ace209465271ed6b8589b494
      drivesStatus: Failed
    - driveID: 3b515b09-3e95-5d25-a583-bee531ff3f31
      driveName: scsi-36000c290ef2632627cb167a03b431a5f
      drivesStatus: Active
    - driveID: 0807fa06-35ce-5a46-9c25-f1669def8c8e
      driveName: scsi-36000c292c8fc037c9f7e97a49e3e2708
      drivesStatus: Active
  ...
```

장애가 발생한 드라이브 CR은 장애가 발생한 드라이브의 UUID에 해당하는 이름으로 클러스터에 자동으로 생성됩니다.

```
kubectl get adsfd -A -o yaml
```

응답:

```
...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSFailedDrive
metadata:
  name: c290a-5000-4652c-9b494
  namespace: astrads-system
spec:
  executeReplace: false
  replaceWith: ""
status:
  cluster: arda-6e4b4af
  failedDriveInfo:
    failureReason: AdminFailed
    inUse: false
    name: scsi-36000c290ace209465271ed6b8589b494
    path: /dev/disk/by-id/scsi-36000c290ace209465271ed6b8589b494
    present: true
    serial: 6000c290ace209465271ed6b8589b494
    node: sti-rx2540-300b.lab.org
  state: ReadyToReplace
```

```
kubectl astrads faileddrive list --cluster arda-6e4b4af
```

응답:

NAME	NODE	CLUSTER	STATE
AGE			
6000c290	sti-rx2540-300b.lab.netapp.com	ard-6e4b4af	ReadyToReplace
13m			

단계

1. 교체 제한 사항에 맞는 드라이브를 필터링하는 "kubbeck astrads faileddrive show-replacement" 명령을 사용하여 가능한 교체 드라이브를 나열합니다(클러스터에서 사용되지 않음, 마운트되지 않음, 파티션 없음, 오류가 발생한 드라이브보다 크거나 같음).

가능한 대체 드라이브를 필터링하지 않고 모든 드라이브를 나열하려면 'show-replacement' 명령에 '--all'을 추가합니다.

```
kubectl astrads faileddrive show-replacements --cluster ard-6e4b4af
--name 6000c290
```

응답:

NAME	IDPATH	SERIAL	PARTITIONCOUNT	MOUNTED	SIZE
sdh	/scsi-36000c29417	45000c	0	false	100GB

2. "replace" 명령을 사용하여 드라이브를 전달된 일련 번호로 교체합니다. 명령이 대체를 완료하거나, '--wait' 시간이 경과되면 실패합니다.

```
kubectl astrads faileddrive replace --cluster arda-6e4b4af --name
6000c290 --replaceWith 45000c --wait
Drive replacement completed successfully
```



부적절한 일련 번호를 사용하여 `kubctl astrads faileddrive replace`를 실행하면 다음과 같은 오류가 나타납니다.

```
kubectl astrads replacedrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith BAD_SERIAL_NUMBER
Drive 6000c2927 replacement started
Failed drive 6000c2927 has been set to use BAD_SERIAL_NUMBER as a
replacement
...
Drive replacement didn't complete within 25 seconds
Current status: {FailedDriveInfo:{InUse:false Present:true Name:scsi-
36000c2 FiretapUUID:444a5468 Serial:6000c Path:/scsi-36000c
FailureReason:AdminFailed Node:sti-b200-0214a.lab.netapp.com}
Cluster:astrads-cluster-f51b10a State:ReadyToReplace
Conditions:[{Message: "Replacement drive serial specified doesn't
exist", Reason: "DriveSelectionFailed", Status: False, Type:' Done'}]}
```

3. 드라이브 교체를 다시 실행하려면 이전 명령으로 '--force'를 사용하십시오.

```
kubectl astrads replacedrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith VALID_SERIAL_NUMBER --force
```

를 참조하십시오

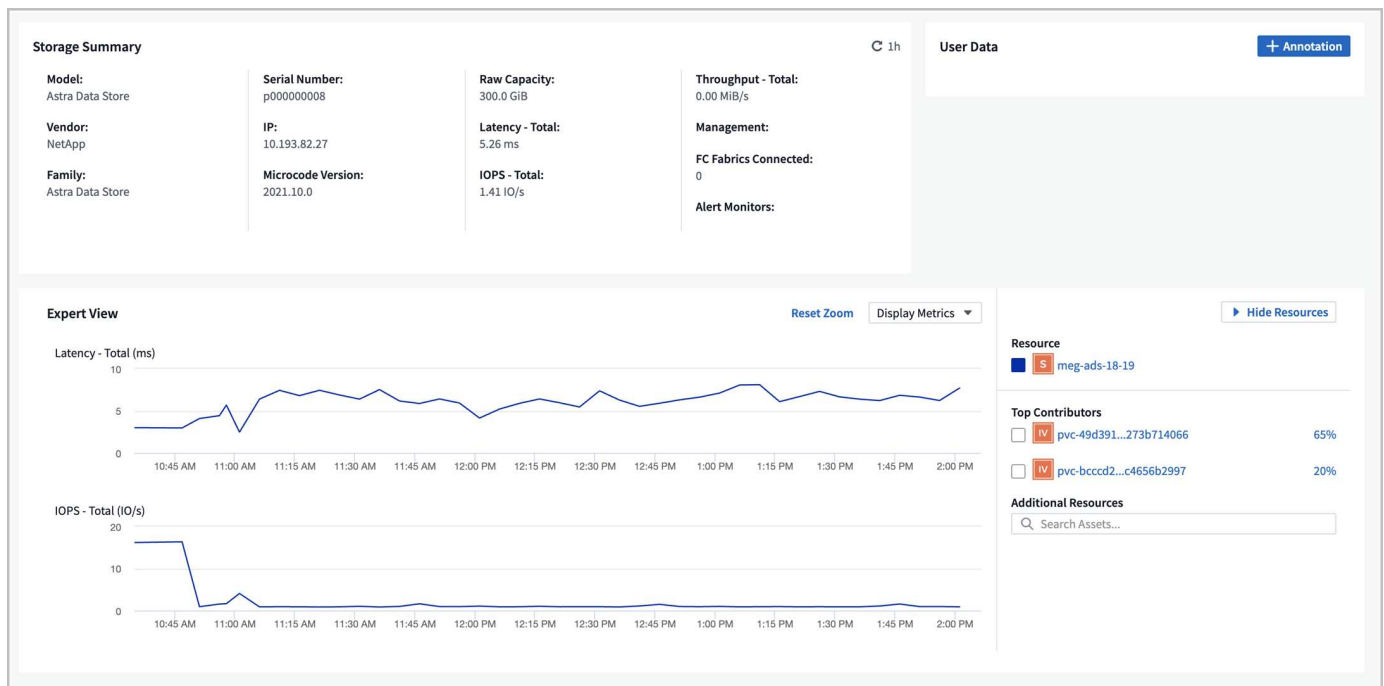
- ["kubect 명령을 사용하여 Astra Data Store 리소스를 관리합니다"](#)

Astra Data Store를 모니터링합니다

Cloud Insights를 통해 메트릭을 모니터링합니다

Cloud Insights를 사용하여 Astra 데이터 저장소 메트릭을 모니터링할 수 있습니다.

다음은 Cloud Insights에 표시되는 몇 가지 Astra 데이터 저장소 메트릭입니다



을 사용하여 Astra Data Store에서 생성된 메트릭 목록을 표시할 수도 있습니다 [\[Open Metrics API help\]](#).

다음 작업을 완료할 수 있습니다.

- [\[Complete Cloud Insights connection prerequisite tasks\]](#)
- [\[Acquisition Unit storage\]](#)
- [\[Download and run the installation script\]](#)
- [\[Edit the Cloud Insights connection\]](#)
- [\[Disconnect from Cloud Insights\]](#)

Cloud Insights 연결 필수 작업을 완료합니다

Astra 데이터 저장소를 Cloud Insights와 연결하기 전에 다음 작업을 완료해야 합니다.

- ["Astra Data Store Monitoring Operator를 설치합니다"](#) 이것은 Astra Data Store 설치 지침의 일부입니다.
- ["kubbeck-astrads 바이너리를 설치합니다"](#) 이것은 Astra Data Store 설치 지침의 일부입니다.
- ["Cloud Insights 계정을 만듭니다"](#).
- 'awk, curl, grep', 'jq' 명령을 사용할 수 있는지 확인합니다

다음 정보를 수집합니다.

- 획득 장치, 데이터 수집, 데이터 수집 및 로그 수집 등의 범주에 대한 읽기/쓰기 권한이 있는 * Cloud Insights API 액세스 토큰 *. 이 작업은 읽기/쓰기 작업, 획득 장치 설정 및 데이터 수집 프로세스 설정에 사용됩니다.
- * Kubernetes API 서버 IP 주소 및 포트 *. Astra Data Store 클러스터를 모니터링하는 데 사용됩니다.
- * Kubernetes API 토큰 *. Kubernetes API를 호출하는 데 사용됩니다.

- * 영구 볼륨 구성 *. 영구 볼륨의 프로비저닝 방법에 대한 정보입니다.

획득 장치 저장

획득 장치는 설치 파일, 구성 데이터 및 로그를 저장하기 위해 세 개의 영구 볼륨이 필요합니다. Monitoring Operator는 기본 스토리지 클래스를 사용하여 영구 볼륨 클레임을 생성합니다. 설치 프로그램 스크립트를 실행할 때 '-s' 옵션을 사용하여 다른 스토리지 클래스 이름을 지정할 수 있습니다.

Kubernetes 클러스터에 스토리지 공급자(예: NetApp Trident)가 없으면 설치 관리자 스크립트를 실행할 때 '-r' 옵션을 사용하여 로컬 파일 시스템 경로를 제공할 수 있습니다. '-r' 옵션이 설정되면 설치 프로그램 스크립트는 제공된 디렉토리 내에 세 개의 영구 볼륨을 생성합니다. 이 디렉토리에는 최소 150GB의 여유 공간이 필요합니다.

설치 스크립트를 다운로드하고 실행합니다

Cloud Insights는 모니터링 오퍼레이터를 통해 Astra 데이터 저장소 모니터링을 활성화하는 Bash 스크립트를 제공합니다. 설치 스크립트는 Astra Data Store Collector와 Fluent Bit Agent를 사용하여 획득 장치를 설치합니다.

Cloud Insights 테넌트 도메인 이름과 선택한 Cloud Insights API 액세스 토큰은 다운로드될 때 설치 프로그램 스크립트에 포함됩니다.

그런 다음 다음과 같이 메트릭이 전송됩니다.

- Cloud Insights 획득 장치는 Cloud Insights 데이터 레이크에 메트릭을 전송합니다.
- Fluent bit는 로그 수집 서비스에 로그를 보냅니다.

설치 프로그램 스크립트 도움말을 표시합니다

설치 프로그램 스크립트에 대한 전체 도움말 텍스트가 아래에 나와 있습니다.

설치 프로그램 스크립트 도움말 텍스트 표시:

```
./cloudinsights-ads-monitoring.sh -h
```

응답:

```

USAGE: cloudinsights-ads-monitoring.sh [OPTIONS]
Configure monitoring of Astra Data Store by Cloud Insights.
OPTIONS:
  -h                                Display this help message.
  -d ci_domain_name                 Cloud Insights tenant domain name.
  -i kubernetes_ip                 Kubernetes API server IP address.
  -k ci_api_key                     Cloud Insights API Access Token.
  -n namespace                       Namespace for monitoring components. (default:
netapp-monitoring)
  -p kubernetes_port                Kubernetes API server port. (default: 6443)
  -r root_pv_dir                    Create 3 Persistent Volumes in this directory
for the Acquisition Unit.
                                     Only specify this option if there is no Storage
Provisioner installed and the PVs do not already exist.
  -s storage_class                  Storage Class name for provisioning Acquisition
Unit PVs. If not specified, the default storage class will be used.
  -t kubernetes_token               Kubernetes API server token.

```


설치 스크립트를 실행합니다

1. Cloud Insights 계정이 없는 경우 계정을 만듭니다.
2. Cloud Insights에 로그인합니다.
3. Cloud Insights 메뉴에서 * Admin * > * Data Collector * 를 클릭합니다.
4. 새 수집기를 추가하려면 * + Data Collector * 를 클릭합니다.




5. Astra Data Store * 타일을 클릭합니다.
6. 올바른 Cloud Insights API 액세스 토큰을 선택하거나 새 토큰을 생성하십시오.
7. 지침에 따라 설치 프로그램 스크립트를 다운로드하고, 권한을 업데이트하고, 스크립트를 실행합니다.


이 스크립트에는 Cloud Insights 테넌트 URL 및 선택한 Cloud Insights API 액세스 토큰이 포함되어 있습니다.



Select a Data Collector



Configure Collector



NetApp
Astra Data Store

Configure Collector

Configure Kubernetes Operator to monitor NetApp Astra Data Store (ADS).

What Operating System or Platform Are You Using?

Kubernetes

Select existing API Access Token or create a new one

default_ads_api_key1 (...d0gHof)

[+ API Access Token](#)

[Production Best Practices ?](#)

Configure Astra Data Store [Need Help?](#)

- 1

The commands *awk*, *curl*, *grep*, *jq*, *kubectl* and the *kubectl-astads* plugin must be installed where the installer script is run. You will need the Kubernetes API server IP address and a Kubernetes API token to run this install script. See the documentation if you need help finding this information.
- 2

Copy Installer Script

☐ Reveal Installer Script

```
#!/usr/bin/env bash
SCRIPT='basename $0'

CI_DOMAIN_NAME="f49uaky.gstabler-ads.cloudinsights-dev.netapp.com"
CI_API_KEY="eyJraWQ1OiI5OTk5IiwidHlwIjoIc3R5bWVudF9hZHNfYXBPX2t1eTEgKG9uIGJlaGFsZiBvZiBhZG1pbik"

```
- 3

Copy the above installer script and save it as *cloudinsights-ads-monitoring.sh*
- 4

Copy Permissions Command

☐ Reveal Permissions Command

```
chmod +x cloudinsights-ads-monitoring.sh

```
- 5

Paste the permissions command in a terminal to enable execute permissions on the installer script.
- 6

Copy Install Command

☐ Reveal Install Command

```
./cloudinsights-ads-monitoring.sh -i <KUBERNETES_IP> -t <KUBERNETES_TOKEN>

```
- 7

Paste the install command in a terminal, replace the placeholders with the correct values for your environment, and run the command. It will take several minutes to complete. The script will use the default namespace 'netapp-monitoring'. Additional options are available for customized environments. The default configuration for kubectl should point to the kubernetes cluster to be monitored.
- 8

Complete Setup

8. 스크립트가 완료되면 * 설치 완료 * 를 클릭합니다.

설치 스크립트가 완료되면 데이터 소스 목록에 Astra Data Store Collector가 나타납니다.



오류로 인해 스크립트가 종료되면 나중에 오류가 해결되면 스크립트를 다시 실행할 수 있습니다. 이 스크립트는 환경에서 기본 설정을 사용하지 않는 경우 모니터링 운영자 네임스페이스 및 Kubernetes API 서버 포트와 같은 추가 매개 변수를 지원합니다. 사용법과 도움말 텍스트를 보려면 의 ``h' 옵션을 사용하십시오./cloudinsights-ads-monitoring.sh -h.

설치 스크립트는 구성이 성공할 때 다음과 같은 출력을 생성합니다.

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Acquisition Unit
...
Acquisition Unit has been installed successfully.
Configuring Astra Data Store data collector
Astra Data Store collector data '<CLUSTER_NAME>' created
Configuration done!
```

상담원 **CR**의 예

다음은 설치 프로그램 스크립트를 실행한 후 Monitoring-NetApp 에이전트 CR이 어떻게 보일지에 대한 예입니다.

```

spec:
  au:
    isEnabled: true
    storageClassName: auto-sc
  cluster-name: meg-ads-21-22-29-30
  docker-repo: docker.repo.eng.netapp.com/global/astra
  fluent-bit:
    - name: ads-tail
      outputs:
        - sink: ADS_STDOUT
      substitutions:
        - key: TAG
          value: firetapems
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
    - name: agent
    - name: ads-tail-ci
      outputs:
        - sink: CI
      substitutions:
        - key: TAG
          value: netapp.ads
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
  output-sink:
    - api-key: abcd
      domain-name: bz19ngz.gst-adsdemo.ci-dev.netapp.com
      name: CI
  serviceAccount: sa-netapp-monitoring
  status:
    au-pod-status: UP
    au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9

```

Cloud Insights 연결을 편집합니다

나중에 Kubernetes API 토큰 또는 Cloud Insights API 액세스 토큰을 편집할 수 있습니다.

- Kubernetes API 토큰을 업데이트하려면 Cloud Insights UI에서 Astra Data Store Collector를 편집해야 합니다.
- 원격 측정 및 로그에 사용되는 Cloud Insights API 액세스 토큰을 업데이트하려면 `kubctl` 명령을 사용하여 모니터링 오퍼레이터 CR을 편집해야 합니다.

Kubernetes API 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. Admin * > * Data Collector * 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 페이지 오른쪽에 있는 메뉴를 클릭하고 * 편집 * 을 선택합니다.
5. Kubernetes API 토큰 필드를 새 값으로 업데이트합니다.
6. Collector 저장 * 을 선택합니다.

Cloud Insights API 액세스 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. 관리자 * > * API 액세스 * 를 선택하고 * + API 액세스 토큰 * 을 클릭하여 새 Cloud Insights API 액세스 토큰을 만듭니다.
3. 상담원 CR 편집:

```
kubect1 --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

4. 출력 싱크 섹션을 찾아 이름이 'CI'인 항목을 찾습니다.
5. 'api-key'라는 레이블의 경우 현재 값을 새 Cloud Insights API 액세스 토큰으로 바꿉니다.

섹션은 다음과 같이 표시됩니다.

```
output-sink:
- api-key: <api key value>
  domain-name: <tenant url>
  name: CI
```

6. 편집기 창을 저장하고 종료합니다.

모니터링 운영자는 새로운 Cloud Insights API 액세스 토큰을 사용하도록 Fluent 비트를 업데이트합니다.

Cloud Insights와의 연결을 해제합니다

Cloud Insights와의 연결을 끊으려면 먼저 Cloud Insights UI에서 Astra 데이터 저장소 수집기를 삭제해야 합니다. 이 작업이 완료되면 모니터링 작동자가 획득 장치, Telegraf(구성된 경우) 및 Fluent 비트 구성을 제거할 수 있습니다.

Astra Data Store Collector를 제거합니다

1. Cloud Insights에 로그인합니다.

2. Admin * > * Data Collector * 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 화면 오른쪽의 메뉴를 선택하고 * Delete * 를 선택합니다.
5. 확인 페이지에서 * 삭제 * 를 클릭합니다.

획득 장치, **Telegraf**(전신)(구성된 경우) 및 **Fluent bit**를 제거합니다

1. 상담원 CR 편집:

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

2. au 섹션을 찾아 IsEnabled를 false로 설정합니다
3. '유창한 비트' 섹션을 찾아 ads-tail-CI'라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 "fluent-bit" 섹션을 제거할 수 있습니다.
4. 텔레그래프가 구성된 경우 텔레그래프 섹션을 찾아 광고 공개 메트릭이라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 Telegraf 섹션을 제거할 수 있습니다.
5. 출력 싱크 섹션을 찾아 'CI'라는 싱크를 제거합니다.
6. 편집기 창을 저장하고 종료합니다.

모니터링 오퍼레이터는 Telegraf(Telegraf)(구성된 경우) 및 Fluent 비트 구성을 업데이트하고 획득 장치 포드를 삭제합니다.

7. Storage Provisioner 대신 Acquisition Unit PVS에 로컬 디렉토리를 사용한 경우 PVS를 삭제합니다.

```
kubectl delete pv au-lib au-log au-pv
```

그런 다음 획득 장치가 실행 중인 노드에서 실제 디렉토리를 삭제합니다.

8. 획득 장치 포드가 삭제된 후 Cloud Insights에서 획득 장치를 삭제할 수 있습니다.
 - a. Cloud Insights 메뉴에서 * Admin * > * Data Collector * 를 선택합니다.
 - b. Acquisition Units(획득 단위) * 탭을 클릭합니다.
 - c. 획득 장치 포드 옆에 있는 메뉴를 클릭합니다.
 - d. 삭제 * 를 선택합니다.

모니터링 오퍼레이터는 Telegraf(전신)(구성된 경우) 및 Fluent bit 구성을 업데이트하고 획득 장치를 제거합니다.

메트릭 **API** 도움말을 엽니다

다음은 Astra Data Store에서 메트릭을 수집하는 데 사용할 수 있는 API 목록입니다.

- "도움말" 줄에 메트릭이 설명되어 있습니다.
- "유형" 선은 메트릭이 게이지 또는 카운터인지 여부를 나타냅니다.

```

# HELP astrads_cluster_capacity_logical_percent Percentage cluster logical
capacity that is used (0-100)
# TYPE astrads_cluster_capacity_logical_percent gauge
# HELP astrads_cluster_capacity_max_logical Max Logical capacity of the
cluster in bytes
# TYPE astrads_cluster_capacity_max_logical gauge
# HELP astrads_cluster_capacity_max_physical The sum of the space in the
cluster in bytes for storing data after provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_max_physical gauge
# HELP astrads_cluster_capacity_ops The IO operations capacity of the
cluster
# TYPE astrads_cluster_capacity_ops gauge
# HELP astrads_cluster_capacity_physical_percent The percentage of cluster
physical capacity that is used (0-100)
# TYPE astrads_cluster_capacity_physical_percent gauge
# HELP astrads_cluster_capacity_used_logical The sum of the bytes of data
in all volumes in the cluster before provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_used_logical gauge
# HELP astrads_cluster_capacity_used_physical Used Physical capacity of a
cluster in bytes
# TYPE astrads_cluster_capacity_used_physical gauge
# HELP astrads_cluster_other_latency The sum of the accumulated latency in
seconds for other IO operations of all the volumes in a cluster. Divide by
astrads_cluster_other_ops to get the average latency per other operation
# TYPE astrads_cluster_other_latency counter
# HELP astrads_cluster_other_ops The sum of the other IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_other_ops counter
# HELP astrads_cluster_read_latency The sum of the accumulated latency in
seconds of read IO operations of all the volumes in a cluster. Divide by
astrads_cluster_read_ops to get the average latency per read operation
# TYPE astrads_cluster_read_latency counter
# HELP astrads_cluster_read_ops The sum of the read IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_read_ops counter
# HELP astrads_cluster_read_throughput The sum of the read throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_read_throughput counter
# HELP astrads_cluster_storage_efficiency Efficacy of data reduction
technologies. (logical used / physical used)
# TYPE astrads_cluster_storage_efficiency gauge
# HELP astrads_cluster_total_latency The sum of the accumulated latency in
seconds of all IO operations of all the volumes in a cluster. Divide by
astrads_cluster_total_ops to get average latency per operation

```



```

# TYPE astrads_cluster_total_latency counter
# HELP astrads_cluster_total_ops The sum of the IO operations of all the
volumes in a cluster
# TYPE astrads_cluster_total_ops counter
# HELP astrads_cluster_total_throughput The sum of the read and write
throughput of all the volumes in a cluster in bytes
# TYPE astrads_cluster_total_throughput counter
# HELP astrads_cluster_utilization_factor The ratio of the current cluster
IO operations based on recent IO sizes to the cluster iops capacity. (0.0
- 1.0)
# TYPE astrads_cluster_utilization_factor gauge
# HELP astrads_cluster_volume_used The sum of used capacity of all the
volumes in a cluster in bytes
# TYPE astrads_cluster_volume_used gauge
# HELP astrads_cluster_write_latency The sum of the accumulated latency in
seconds of write IO operations of all the volumes in a cluster. Divide by
astrads_cluster_write_ops to get the average latency per write operation
# TYPE astrads_cluster_write_latency counter
# HELP astrads_cluster_write_ops The sum of the write IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_write_ops counter
# HELP astrads_cluster_write_throughput The sum of the write throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_write_throughput counter
# HELP astrads_disk_base_seconds Base for busy, pending and queued.
Seconds since collection began
# TYPE astrads_disk_base_seconds counter
# HELP astrads_disk_busy Seconds the disk was busy. 100 *
(astrads_disk_busy / astrads_disk_base_seconds) = percent busy (0-100)
# TYPE astrads_disk_busy counter
# HELP astrads_disk_capacity Raw Capacity of a disk in bytes
# TYPE astrads_disk_capacity gauge
# HELP astrads_disk_io_pending Summation of the count of pending io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average pending operation count
# TYPE astrads_disk_io_pending counter
# HELP astrads_disk_io_queued Summation of the count of queued io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average queued operations count
# TYPE astrads_disk_io_queued counter
# HELP astrads_disk_read_latency Total accumulated latency in seconds for
disk reads. Divide by astrads_disk_read_ops to get the average latency per
read operation
# TYPE astrads_disk_read_latency counter
# HELP astrads_disk_read_ops Total number of read operations for a disk
# TYPE astrads_disk_read_ops counter

```

```

# HELP astrads_disk_read_throughput Total bytes read from a disk
# TYPE astrads_disk_read_throughput counter
# HELP astrads_disk_write_latency Total accumulated latency in seconds for
disk writes. Divide by astrads_disk_write_ops to get the average latency
per write operation
# TYPE astrads_disk_write_latency counter
# HELP astrads_disk_write_ops Total number of write operations for a disk
# TYPE astrads_disk_write_ops counter
# HELP astrads_disk_write_throughput Total bytes written to a disk
# TYPE astrads_disk_write_throughput counter
# HELP astrads_value_scrape_duration Duration to scrape values
# TYPE astrads_value_scrape_duration gauge
# HELP astrads_volume_capacity_available The minimum of the available
capacity of a volume and the available capacity of the cluster in bytes
# TYPE astrads_volume_capacity_available gauge
# HELP astrads_volume_capacity_available_logical Logical available
capacity of a volume in bytes
# TYPE astrads_volume_capacity_available_logical gauge
# HELP astrads_volume_capacity_percent Percentage of volume capacity
available (0-100). (capacity available / provisioned) * 100
# TYPE astrads_volume_capacity_percent gauge
# HELP astrads_volume_capacity_provisioned Provisioned capacity of a
volume in bytes after setting aside the snapshot reserve. (size - snapshot
reserve = provisioned)
# TYPE astrads_volume_capacity_provisioned gauge
# HELP astrads_volume_capacity_size Total capacity of a volume in bytes
# TYPE astrads_volume_capacity_size gauge
# HELP astrads_volume_capacity_snapshot_reserve_percent Snapshot reserve
percentage of a volume (0-100)
# TYPE astrads_volume_capacity_snapshot_reserve_percent gauge
# HELP astrads_volume_capacity_snapshot_used The amount of volume snapshot
data that is not in the active file system in bytes
# TYPE astrads_volume_capacity_snapshot_used gauge
# HELP astrads_volume_capacity_used Used capacity of a volume in bytes.
This is bytes in the active filesystem unless snapshots are consuming more
than the snapshot reserve. (bytes in the active file system + MAX(0,
snapshot_used-(snapshot_reserve_percent/100*size))
# TYPE astrads_volume_capacity_used gauge
# HELP astrads_volume_other_latency Total accumulated latency in seconds
for operations on a volume that are neither read or write. Divide by
astrads_volume_other_ops to get the average latency per other operation
# TYPE astrads_volume_other_latency counter
# HELP astrads_volume_other_ops Total number of operations for a volume
that are neither read or write
# TYPE astrads_volume_other_ops counter
# HELP astrads_volume_read_latency Total accumulated read latency in

```

```
seconds for a volume. Divide by astrads_volume_read_ops to get the average
latency per read operation
# TYPE astrads_volume_read_latency counter
# HELP astrads_volume_read_ops Total number of read operations for a
volume
# TYPE astrads_volume_read_ops counter
# HELP astrads_volume_read_throughput Total read throughput for a volume
in bytes
# TYPE astrads_volume_read_throughput counter
# HELP astrads_volume_total_latency Total accumulated latency in seconds
for all operations on a volume. Divide by astrads_volume_total_ops to get
the average latency per operation
# TYPE astrads_volume_total_latency counter
# HELP astrads_volume_total_ops Total number of operations for a volume
# TYPE astrads_volume_total_ops counter
# HELP astrads_volume_total_throughput Total throughput for a volume in
bytes
# TYPE astrads_volume_total_throughput counter
# HELP astrads_volume_write_latency Total accumulated write latency in
seconds for volume. Divide by astrads_volume_write_ops to get the average
latency per write operation
# TYPE astrads_volume_write_latency counter
# HELP astrads_volume_write_ops Total number of write operations for a
volume
# TYPE astrads_volume_write_ops counter
# HELP astrads_volume_write_throughput Total write throughput for a volume
in bytes
# TYPE astrads_volume_write_throughput counter
```

Prometheus 및 Grafana를 사용하여 메트릭을 모니터링합니다

Prometheus 및 Grafana를 사용하여 Astra Data Store 메트릭을 모니터링할 수 있습니다. Astra Data Store Kubernetes 클러스터 메트릭 엔드포인트에서 메트릭을 수집하도록 Prometheus를 구성할 수 있으며 Grafana를 사용하여 메트릭 데이터를 시각화할 수 있습니다.

무엇을 '필요로 할거야

- Astra Data Store 클러스터 또는 Astra Data Store 클러스터와 통신할 수 있는 다른 클러스터에 Prometheus 및 Grafana 패키지를 다운로드하여 설치했는지 확인하십시오. 공식 설명서의 지침에 따라 각 도구를 설치합니다.
 - ["Prometheus를 설치합니다"](#)
 - ["Grafana를 설치합니다"](#)
- Prometheus 및 Grafana는 Astra Data Store Kubernetes 클러스터와 통신할 수 있어야 합니다. Prometheus 및 Grafana가 Astra Data Store 클러스터에 설치되어 있지 않은 경우, Astra Data Store 클러스터에서 실행 중인 메트릭 서비스와 통신할 수 있어야 합니다.

Prometheus를 구성합니다

Astra Data Store는 Kubernetes 클러스터의 TCP 포트 9341에 메트릭 서비스를 제공합니다. 이 서비스에서 메트릭을 수집하려면 Prometheus를 구성해야 합니다.

단계

1. Prometheus 설치를 위해 'Prometheus.yml' 설정 파일을 편집합니다.
2. Astra Data Store 서비스 이름과 해당 포트를 가리키는 서비스 목표를 추가합니다. 예를 들면 다음과 같습니다.

```
scrape_configs:
static_configs:
- targets: ['astrads-metrics-service.astrads-system:9341']
```

3. Prometheus 서비스를 시작합니다.

Grafana 구성

Grafana를 구성하여 Prometheus에서 수집한 메트릭을 표시할 수 있습니다.

단계

1. Grafana 설치를 위해 'datasources.yaml' 구성 파일을 편집합니다.
2. Prometheus를 데이터 소스로 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: 1

datasources:
- name: astradatastore-prometheus
  type: prometheus
  access: proxy
  url: http://localhost:9090
  jsonData:
    manageAlerts: false
```

3. Grafana 서비스를 시작합니다.
4. 에 대한 Grafana 설명서의 지침을 따릅니다 "[시작하십시오](#)".

Grafana 대시보드 템플릿을 가져옵니다

Astra Data Store를 설치하기 위해 다운로드한 번들 파일에는 Grafana 내에서 가져올 수 있는 Grafana 대시보드 템플릿 파일이 포함되어 있습니다. 이러한 대시보드 템플릿을 사용하면 Astra Data Store에서 사용 가능한 메트릭의 유형과 이를 보는 방법을 확인할 수 있습니다.

단계

1. Astra Data Store의 .tar.gz bundle을 엽니다.
2. 'manifests' 디렉터리를 엽니다.

3. graana_cluster.json과 graana_volume.json 파일을 추출한다.
4. Grafana 웹 UI 사용, "[에서 대시보드 템플릿 파일을 Grafana로 가져옵니다](#)".

이벤트 로그를 구성하고 모니터링합니다

EMS(이벤트 관리 시스템) 로그를 모니터링하려면 다음과 같은 높은 수준의 작업을 수행해야 합니다.

- [\[Configure monitoring in the Astra Data Store cluster custom resource \(CR\)\]](#)
- [\[Set up Cloud Insights\]](#)
- [\[Stream event logs to Elastic\]](#).

Astra Data Store 클러스터 사용자 지정 리소스(CR)에서 모니터링 구성

Astra Data Store 클러스터 CR에 모니터링 옵션이 구성되어 있지 않은 경우 "Astrads" 확장을 사용하여 설정할 수 있습니다.

입력:

```
kubectl astrads monitoring setup -n <NAMESPACE OF AGENT INSTALLED> -r  
<DOCKER REPO TO FIND FLUENT/TELEGRAF ETC IMAGES>
```

여기서,

- 설치된 에이전트의 네임스페이스: 모니터링 에이전트의 기본 이름인 모니터링 에이전트의 네임스페이스를 입력합니다. 모니터링 오퍼레이터의 경우 NetApp CR입니다.
- '-r'은 Fluent 또는 Telegraf 이미지가 있는 Docker 레지스트리를 설정하는 데 선택 사항입니다. 기본적으로 경로는 변경할 수 있는 `dddocker.repo.eng.netapp.com/global/astra`` 로 설정됩니다.

Cloud Insights를 설정합니다

로그를 보려면 Cloud Insights 설정은 선택 사항이지만 Cloud Insights를 사용하여 데이터를 보는 것이 좋습니다. 을 참조하십시오 "[NetApp Cloud Insights 설정 방법](#)" Astra Data Store와 함께 사용

이벤트 로그를 **Elastic**로 스트리밍합니다

EMS 이벤트 및 기타 POD 로그를 Elastic 등의 타사 엔드포인트로 스트리밍하려면 "Astrads" 확장을 사용합니다.

입력:

```
kubectl astrads monitoring --host <ELASTIC HOST NAME> --port <ELASTIC HOST  
PORT> es
```



탄성 호스트 이름은 IP 주소일 수 있습니다.

보안 Astra 데이터 저장소

보안 인증서를 관리합니다

Astra Data Store는 클러스터의 소프트웨어 구성 요소 간에 MTL(Mutual Transport Layer Security) 암호화를 사용합니다. 각 Astra Data Store 클러스터에는 자체 서명된 루트 CA 인증서("astrads-cert-root")와 중간 CA 인증서("astrads-cert-<cluster_name>")가 있습니다. 이 인증서는 Astra Data Store 운영자가 관리합니다. 운영자는 만료 날짜 7일 전에 각 인증서를 자동으로 갱신합니다. 인증서를 수동으로 취소할 수도 있습니다.

인증서를 해지합니다

Astra Data Store 컨트롤러, 노드 또는 CA 인증서가 손상된 경우 MTL 암호를 삭제하여 이를 취소할 수 있습니다. 이렇게 하면 Astra Data Store 운영자가 자동으로 새 인증서를 발급합니다. 언제든지 Astra Data Store 인증서를 해지할 수 있습니다.



CA 인증서를 해지하면 해당 CA에서 서명한 인증서가 해지됩니다.

단계

1. Astra Data Store 클러스터의 컨트롤러 노드에 로그인합니다.
2. 시스템에 있는 기존 인증서를 나열합니다. 예를 들면 다음과 같습니다.

```
kubectl get secrets -n astrads-system | grep astrads-cert
```

출력은 다음과 비슷해야 합니다.

```
astrads-cert-astrads-cluster-controller
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-dms-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-root
kubernetes.io/tls      4      6d6h
astrads-cert-sti-net-com
kubernetes.io/tls      5      6d6h
```

3. 출력에서 취소할 인증서의 이름을 기록합니다.
4. kubectl 유틸리티를 사용하여 인증서를 해지하고 certificate_name을 인증서 이름으로 바꿉니다. 예를 들면 다음과 같습니다.

```
kubectl delete secret CERTIFICATE_NAME -n astrads-system
```

기존 인증서가 해지되고 대신 새 인증서가 자동으로 생성됩니다.

외부 키 관리

하나 이상의 외부 키 관리 서버를 사용하여 클러스터가 암호화된 데이터에 액세스하는 데 사용하는 키를 보호할 수 있습니다. 외부 키 관리 서버는 KMIP(Key Management Interoperability Protocol)를 사용하여 노드에 키를 제공하는 스토리지 환경의 타사 시스템입니다.



Astra Data Store는 Astra Data Store 클러스터를 생성할 때 기본적으로 내부 키 제공업체와 함께 Software Encryption at Rest(sear)를 활성화합니다.

키 관리에는 다음과 같은 사용자 정의 리소스 정의(CRD)가 포함됩니다.

- *** AstraDSKeyProvider ***: 외부 KMIP 서버를 구성합니다. 이는 서버 클러스터일 수 있습니다.
- *** AstraDSSEARKeyRotate ***: 키 공급자에서 새 키 암호화 키를 가져옵니다. 이 키를 Astra Data Store에 제공합니다.

외부 키 관리와 관련된 다음 작업을 수행할 수 있습니다.

- [\[Set up external key management\]](#)
- [\[Check the software encryption at rest status\]](#)
- [\[Change external to internal key management\]](#)
- [\[Rotate keys for security\]](#)

외부 키 관리를 설정합니다

Astra Data Store에서 외부 키 관리를 설정하는 것은 kubectl asts 명령어를 사용한다.

예를 들어, OpenSSL을 사용하여 외부 키를 설정할 수 있도록 클러스터 또는 KMIP 서버에 SSL 인증서가 필요합니다.

단계

1. 키 공급자 클라이언트에 대한 인증서를 준비합니다. 클라이언트 인증서, 클라이언트 개인 키 및 신뢰 CA 번들을 포함합니다.



예를 들어, OpenSSL을 사용하여 외부 키를 설정할 수 있도록 클러스터 또는 KMIP 서버에 SSL 인증서를 준비합니다.

2. Astra Data Store 클러스터의 노드 중 하나에 로그인합니다.
3. 다음 kubectl extension 명령을 입력하여 Astra Data Store 클러스터의 주요 공급자를 구성합니다.

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=<kmip_server_ip> <key_provider_cr_name>
--namespace astrads-system --cluster <ads_cluster_name>
```

다음 예에서는 ADS 클러스터 "astrads-cluster-f23d158"에 대해 "hashicorp"라는 외부 키 공급자를 구성합니다.

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=10.235.nnn.nnn hashicorp
--namespace astrads-system --cluster astrads-cluster-f23d158
```

1. AstraDSCluster CR을 통해 sear에 외부 키 관리자를 사용하도록 Astra Data Store 클러스터를 구성합니다. 도움말을 표시합니다.

```
kubectl-astrads clusters sears -h
```

응답:

Configure SEARS in AstraDS cluster

Usage:

```
astrads clusters sears [flags]
```

Flags:

```
-d, --duration string    Duration for key rotation (default "2160h")
-h, --help               help for sears
```

Global Flags:

```
--ads-cluster-name string    Name of the ADS Cluster
--ads-cluster-namespace string Namespace of the ADS Cluster
...
```

다음 명령을 실행하면 Astra Data Store 클러스터가 sear의 Key Manager로 "AstraDSKeyProvider hashicorp"를 사용하도록 구성됩니다. 이 명령은 또한 기본값 90일(2160시간)의 키 회전 시간을 사용합니다.

```
kubectl-astrads clusters sears -d 500h hashicorp
--ads-cluster-name=astrads-cluster-f23d158
--ads-cluster-namespace=astrads-system
```


소프트웨어 암호화 유효 상태를 확인합니다

저장된 소프트웨어 암호화 구성을 확인할 수 있습니다.

단계

1. AstraDSCluster CR을 검사합니다.

```
Name:          astrads-cluster-f23d158
Namespace:     astrads-system
Labels:        <none>
Annotations:   <none>
API Version:   astrads.netapp.io/v1beta1
Kind:          AstraDSCluster
...
Spec:
...
  Software Encryption At Rest:
    Ads Key Provider:      hashicorp
    Key Rotation Period:   500h0m0s
...
Status:
...
  Software Encryption At Rest Status:
    Key Active Time:       2022-05-16T15:53:47Z
    Key Provider Name:     hashicorp
    Key Provider UUID:     ccfc2b0b-dd98-5ca4-b778-99debef83550
    Key UUID:              nnnnnnnnnn-nnnnn-nnnnn-nnnnn-nnnnnnnnnnnnnnn
```

내부 키 관리를 외부로 변경합니다

현재 외부 키 관리자를 사용 중인 경우 내부 키 관리자로 변경할 수 있습니다.

단계

1. SoftwareEncryptionAtRest 구성을 제거하여 AstraDSCluster CR을 변경합니다.
2. (선택 사항) 이전 AstraDSKeyProvider 및 관련 암호를 삭제합니다.



이전 키 공급자와 암호는 자동으로 제거되지 않습니다.

보안을 위해 키를 회전합니다

키 로테이션을 통해 보안이 강화됩니다. 기본적으로 Astra Data Store는 90일마다 자동으로 키를 순환합니다. 기본 설정을 변경할 수 있습니다. 또한 필요할 때 키를 회전할 수도 있습니다.

자동 키 회전을 구성합니다

1. CRD에서 AstraDSSEARKeyRotate 매개변수를 업데이트합니다.

```
kubectl patch astradscluster astrads-cluster-f23d158
-n astrads-system
--type=merge -p '{"spec": {"softwareEncryptionAtRest": {
"keyRotationPeriod": "3000h"}}}'
```

주문형 키 회전을 구성합니다

1. 키를 회전하기 위해 AstraDSSEARKeyRotateRequest CR을 생성합니다.

```
cat << EOF | kubectl apply -f -
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSSEARKeyRotateRequest
metadata:
  name: manual
  namespace: astrads-system
spec:
  cluster: astrads-cluster-f23d158
EOF
```

Astra Data Store 라이선스를 업데이트합니다

평가 기간을 연장하기 위해 Astra Data Store에 대해 설치된 평가 라이선스를 업데이트할 수 있습니다. 다음 세 가지 방법 중 하나를 사용하여 라이선스를 업데이트할 수 있습니다.

- Astra Control Center를 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 ["스토리지 백엔드 라이선스를 업데이트합니다"](#).
- Astra VMware 플러그인을 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 ["VMware를 사용하여 Astra Data Store를 관리합니다"](#).
- 명령줄을 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 [\[Update the Astra Data Store license using the command line\]](#).

명령줄을 사용하여 Astra Data Store 라이선스를 업데이트합니다

kubeck 유틸리티를 사용하여 Astra Data Store 라이선스를 업데이트할 수 있습니다.

단계

1. NetApp에서 구입한 대체 NetApp 라이선스 파일(NLF)을 적용합니다. 명령을 실행하기 전에 클러스터 이름(<<Astra-Data-Store-cluster-name>>)과 라이선스 파일 경로('<file_path/file.txt>')를 입력합니다.

```
kubectl astrads license add --license-file-path <file_path/file.txt>
--ads-cluster-name <Astra-Data-Store-cluster-name> -n astrads-system
```

2. 라이선스가 추가되었는지 확인합니다.

```
kubectl astrads license list
```

다음과 유사한 응답이 표시됩니다.

NAME	ADSCLUSTER	VALID	PRODUCT
EVALUATION	ENDDATE	VALIDATED	
p1000000006	astrads-example-cluster	true	Astra Data Store
2023-01-23	2022-04-04T14:38:54Z		true

Astra Data Store를 업그레이드합니다

Astra Data Store를 업그레이드하여 최신 기능과 수정 사항을 활용할 수 있습니다. Astra Data Store의 kubeck 확장자로 Astra Data Store를 업그레이드할 수 있습니다.

kubectl을 사용하여 Astra Data Store를 업그레이드합니다

Astra Data Store의 kubectl을 이용하여 Astra Data Store를 업그레이드할 수 있습니다.

Astra Data Store 번들을 다운로드하고 이미지를 추출합니다

단계

1. 에 로그인합니다 "[NetApp Support 사이트](#)" Astra Data Store 번들('Astra_Data_Store_2022.05.tar')을 다운로드하십시오.
2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify Astra_Data_Store_2022.05.pub -signature  
Astra_Data_Store_2022.05.sig 2022.12.01_ads.tar
```

3. 디렉토리 생성:

```
mkdir Astra_Data_Store_2022.05  
cd Astra_Data_Store_2022.05
```

4. 이미지 추출:

```
tar -vxzf <path to tar file>/Astra_Data_Store_2022.05.tar
```



이미지는 작업 디렉토리 내에 생성된 'Astrads/images' 디렉토리에 압축이 풀립니다.

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

단계

1. 쿠버네티스 쿠버네티스 바이너리가 설치된 표준 경로에 이미지를 추출하는 데 사용한 디렉토리에서 kubectl-astrs 바이너리를 복사합니다("/usr/bin"은 아래 예에서 경로로 사용됨). kubectl-astra 데이터 저장소 클러스터를 설치 및 관리하는 사용자 지정 kubectl 확장입니다.



kubbeck 명령을 사용하여 kubectl 바이너리가 설치된 경로를 찾습니다.

```
cp -p .astrads/bin/kubectl-astrads /usr/bin/.
```

2. Astra Data Store 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.



아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

- a. 레지스트리에 로그인합니다.

```
docker login [your_registry_path]
```

- b. 환경 변수를 레지스트리 경로로 설정하여 Astra Data Store 이미지를 푸시합니다(예: REpo.company.com).

```
export REGISTRY=repo.company.com/astrads
```

- c. 다음 스크립트를 실행하여 이미지를 Docker에 로드하고, 이미지에 태그를 지정한 다음 이미지를 로컬 레지스트리에 푸시합니다.

```
for astraImageFile in $(ls astrads/images/*.tar) ; do
    astraImage=$(docker load --input ${astraImageFile} | sed 's~Loaded
image(s): ~~')
    astraImageShort=`echo $astraImage | sed 's~.*/~~'`
    docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
    docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR_REGISTRY\]~'${REGISTRY}'~'
./astrads/manifests/*.yaml
```

업그레이드를 수행합니다

단계

1. Astradsoperator.YAML 파일을 로컬 디렉토리에 복사합니다.

```
cp /PATH/TO/FILE/astradsoperator.yaml ./
```

2. 운영자를 업그레이드 한다. 대문자의 인수를 환경에 적합한 정보로 바꿉니다.

```
kubectl-astrads upgrade ads-operator --repository-url REPOSITORY_URL  
--operator-yaml astradsoperator.yaml
```

3. Astra Data Store 업그레이드를 시작합니다. 대문자의 인수를 환경에 적합한 정보로 바꿉니다.

```
kubectl-astrads upgrade ads-version --repository-url REPOSITORY_URL  
--ads-version-yaml ./astrads/manifests/astradsversion.yaml
```

업그레이드가 시작되었다는 메시지가 나타나고 완료하는 데 몇 분 정도 걸립니다.

자동 스크립트로 Astra Data Store를 제거합니다

Astra Data Store와 제어 플레인을 제거하려면 워크로드, 바인딩, 볼륨, 내보내기 정책, Astra Data Store 클러스터, 라이선스, 배포 환경 및 Astra Data Store 네임스페이스를 제거합니다.

설치 제거 방법은 다음과 같습니다.

- [\[Uninstall Astra Data Store with an automated script\]](#)
- [\[Uninstall Astra Data Store manually without a script\]](#)
- [\[Troubleshoot the Astra Data Store uninstall process\]](#)

자동 스크립트로 Astra Data Store를 제거합니다

이 프로세스는 자동 스크립트를 사용하여 Astra Data Store를 제거합니다.

무엇을 '필요로 할거야

- 루트 관리 권한

Astra Data Store 제거 프로세스는 다음과 같은 고급 단계를 안내합니다.

- [\[Remove existing workloads and bindings\]](#)
- [\[Uninstall Astra Data Store cluster\]](#)
- [\[Validate the removal of the astrads-system namespace\]](#)
- [\[Ensure containers are not running on worker nodes\]](#)
- [\[Delete OpenShift Container Platform resources\]](#)

기존 워크로드 및 바인딩을 제거합니다

Astra Data Store를 제거하기 전에 먼저 다음을 제거해야 합니다

- Astra Data Store를 스토리지 백엔드로 사용하는 모든 애플리케이션 워크로드
- Astra Data Store를 백엔드로 사용하는 Trident 바인딩

이렇게 하면 Kubernetes 환경이 클린 상태로 유지됩니다. 이는 다시 설치할 때 중요합니다.

Astra Data Store 클러스터를 제거합니다

Astra Data Store를 설치 제거하려면 NetApp Support 사이트에서 다운로드한 Astra Data Store tar 파일에 있는 `uninstall.sh` 스크립트를 사용하십시오.

1. 'manifests' 디렉토리에서 `uninstall.sh`를 찾습니다.
2. 다음 'ed' 명령을 실행합니다.

```
sed -i -e 's~netappsdsoperator.yaml~astradsoperator.yaml~' uninstall.sh
```

3. 제거할 항목을 나타내는 다음 스크립트를 실행합니다.

```
./uninstall.sh

You must run this script with an argument specifying what should be
uninstalled
To uninstall the ADS cluster run ./uninstall.sh cluster
To uninstall everything run ./uninstall all
```

4. 클러스터를 설치 제거하려면 "`uninstall.sh <cluster>`"를 입력하십시오

그렇지 않으면 모든 항목을 제거하려면 `uninstall.sh`를 입력합니다



대부분의 경우 모든 항목을 제거합니다. 클러스터를 나중에 다시 배포하려는 경우 클러스터만 제거하면 됩니다.

5. 프롬프트에서 계속 진행할지 확인하고 `erasedata`를 입력합니다

응답:

```
./uninstall.sh all

Enter 'erasedata' to confirm you want proceed with the uninstall:
erasedata
+-----+
| Wed Feb  2 10:14:01 EST 2022                               |
| ADS cluster uninstall started                               |
+-----+
Deleting astradsvolumes
Deleted astradsvolumes
```

```

Deleting astradsexportpolicies
Deleted astradsexportpolicies
Deleting astradsvolumesnapshots
Deleted astradsvolumesnapshots
Deleting astradsclusters
Deleting astradsclusters
Deleting astradslicenses
Deleted astradslicenses

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS cluster uninstall done   |
+-----+

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS system uninstall started  |
+-----+

Removing astradsversion
astradsversion.astrads.netapp.io "astradsversion" deleted
Removed astradsversion
Removing daemonsets
daemonset.apps "astrads-ds-nodeinfo-astradsversion" deleted
Removed daemonsets
Removing deployments
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted
Removed deployments
Removing all other AstraDS resources
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscloudsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsoptions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io

```

```
"astradsnodeinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodemanagements.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqospolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsversions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumefiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumesnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-admin-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-reader-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-writer-role"
deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
role.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-admin-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-reader-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-writer-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
```



```
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsqospolicy-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
```

```
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-
rolebinding" deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
secret "astrads-autosupport-certs" deleted
+-----+
| Wed Feb  2 10:16:36 EST 2022 |
| ADS system uninstall done   |
+-----+
```

Astrads-system namespace의 제거를 검증합니다

다음 명령을 실행하면 결과가 반환되지 않는지 확인합니다.

```
kubectl get ns | grep astrads-system
```

컨테이너를 작업자 노드에서 실행하지 않도록 합니다

소방관이나 netwd 같은 컨테이너가 작업자 노드에서 실행되고 있지 않는지 확인합니다. 각 노드에서 다음을 실행합니다.

```
ssh <mynode1>
# runc list
```

OpenShift Container Platform 리소스를 삭제합니다

Red Hat OpenShift Container Platform(OCP)에 Astra Data Store를 설치한 경우 OCP SCC(Security Context Constraints) 및 rolebindings 리소스를 제거할 수 있습니다.

OpenShift는 POD가 수행할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 사용합니다.

표준 제거 프로세스를 완료한 후 다음 단계를 완료합니다.

1. SCC 리소스 제거:

```
oc delete -f ads_privileged_scc.yaml
```

2. rolebindings 리소스 제거:

```
oc delete -f oc_role_bindings.yaml
```



이 단계에서는 "리소스를 찾을 수 없음" 오류를 무시합니다.

스크립트 없이 **Astra Data Store**를 수동으로 제거합니다

이 프로세스는 스크립트 없이 Astra Data Store를 수동으로 제거합니다.

자동 스크립트 없이 Astra Data Store를 수동으로 제거하려면 워크로드, 바인딩, 볼륨, 익스포트 정책, 클러스터, 라이선스, 배포 환경 및 Astra Data Store 네임스페이스.

무엇을 '필요로 할거야

- 루트 관리 권한

Astra Data Store 제거 프로세스는 다음과 같은 고급 단계를 안내합니다.

- [\[Remove existing workloads and bindings\]](#)
- [\[Uninstall the Astra Data Store cluster and control plane\]](#)
- [\[Delete the license\]](#)
- [\[Delete the Astra Data Store installation\]](#)
- [\[Validate the removal of the astrads-system namespace\]](#)
- [\[Ensure containers are not running on worker nodes\]](#)
- [\[Delete OpenShift Container Platform resources\]](#)

기존 워크로드 및 바인딩을 제거합니다

Astra Data Store를 제거하기 전에 먼저 다음을 제거해야 합니다

- Astra Data Store를 스토리지 백엔드로 사용하는 모든 애플리케이션 워크로드
- Astra Data Store를 백엔드로 사용하는 Trident 바인딩

이렇게 하면 Kubernetes 환경이 클린 상태로 유지됩니다. 이는 다시 설치할 때 중요합니다.

Astra Data Store 클러스터와 컨트롤 플레인을 제거합니다

Astra Data Store를 수동으로 제거하려면 아래 단계를 따르십시오.

볼륨 및 익스포트 정책을 삭제합니다

클러스터를 삭제하기 전에 Astra Data Store 볼륨과 익스포트 정책을 삭제해야 합니다.



먼저 볼륨 및 익스포트 정책을 삭제하지 않으면 Astra Data Store 볼륨 객체가 삭제될 때까지 클러스터 삭제 프로세스가 일시 중지됩니다. 클러스터 삭제를 시작하기 전에 이러한 항목을 제거하는 것이 더 효율적입니다.

단계

1. 볼륨 삭제:

```
~% kubectl delete astradsvolumes --all -A
~% kubectl get astradsvolumes -A
```

2. 익스포트 정책 삭제:

```
~% kubectl delete astradsexportpolicies --all -A
~% kubectl get astradsexportpolicies -A
```

Astra Data Store 클러스터를 삭제합니다

클러스터를 삭제하면 클러스터 범위 리소스와 함께 Astra CR(Data Store Cluster Object Custom Resource)만 삭제됩니다.



클러스터가 삭제된 후에도 운영자, 노드 정보 포드 및 클러스터 컨트롤러(Kubernetes 범위 리소스)는 그대로 유지됩니다.

클러스터를 삭제하면 기본 운영 체제가 노드에서 제거되므로 'firtap' 및 'netwd' 서비스가 중지됩니다.

제거 프로그램을 완료하는 데 약 1분 정도 걸립니다. 그런 다음 Astra Data Store 클러스터 범위 리소스 제거가 시작됩니다.

1. 클러스터를 삭제합니다.

```
~% kubectl delete astradsclusters --all -A
~% kubectl get astradsclusters -A
```

라이센스를 삭제합니다

1. 클러스터의 각 작업자 노드에 SSH를 통해 'firewTap' 또는 'netwd'가 작업자 노드에서 실행되고 있지 않음을 확인합니다.
2. Astra Data Store 라이선스를 삭제합니다.

```
~% kubectl delete astradslicenses --all -A
~% kubectl get astradslicenses -A
```

Astra Data Store 설치를 삭제합니다

클러스터에서 컨트롤러, 운영자, 네임스페이스 및 지원 포드를 삭제하십시오.

1. Astra Data Store 설치 객체를 삭제합니다.

```
~% kubectl delete astradsversion astradsversion -n astrads-system
~% kubectl get astradsversion -n astrads-system
```

2. 데이터 저장소 DemonSets 및 모든 Astra Data Store 컨트롤러 리소스를 삭제합니다.

```
~% kubectl delete ds --all -n astrads-system
~% kubectl get ds -n astrads-system

~% kubectl delete deployments --all -n astrads-system
~% kubectl get deployments -n astrads-system
```

3. 나머지 아티팩트 및 작업자 YAML 파일 삭제:

```
~% kubectl delete -f ./manifests/astradsoperator.yaml
~% kubectl get pods -n astrads-system
```

Astrads-system namespace의 제거를 검증합니다

다음 명령을 실행하면 결과가 반환되지 않는지 확인합니다.

```
~% kubectl get ns | grep astrads-system
```

컨테이너를 작업자 노드에서 실행하지 않도록 합니다

소방관이나 netwd 같은 컨테이너가 작업자 노드에서 실행되고 있지 않는지 확인합니다. 각 노드에서 다음을 실행합니다.

```
ssh <mynode1>
# runc list
```

OpenShift Container Platform 리소스를 삭제합니다

Red Hat OpenShift Container Platform(OCP)에 Astra Data Store를 설치한 경우 OCP SCC(Security Context Constraints) 및 rolebindings 리소스를 제거할 수 있습니다.

OpenShift는 POD가 수행할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 사용합니다.

표준 제거 프로세스를 완료한 후 다음 단계를 완료합니다.

1. SCC 리소스 제거:

```
oc delete -f ads_privileged_scc.yaml
```

2. rolebindings 리소스 제거:

```
oc delete -f oc_role_bindings.yaml
```



이 단계에서는 "리소스를 찾을 수 없는 오류"를 무시합니다.

수동 삭제 샘플

다음은 실행 수동 제거 스크립트의 예입니다.

```
$ kubectl delete astradsvolumes --all -A
No resources found
$ kubectl delete astradsexportpolicies --all -A
No resources found
$ kubectl delete astradsclusters --all -A
astradscluster.astrads.netapp.io "astrads-sti-c6220-09-10-11-12" deleted

$ kubectl delete astradslicenses --all -A
astradslicense.astrads.netapp.io "e900000005" deleted

$ kubectl delete astradsdeployment astradsdeployment -n astrads-system
astradsdeployment.astrads.netapp.io "astradsdeployment" deleted

$ kubectl delete ds --all -n astrads-system
daemonset.apps "astrads-ds-astrads-sti-c6220-09-10-11-12" deleted
daemonset.apps "astrads-ds-nodeinfo-astradsdeployment" deleted
daemonset.apps "astrads-ds-support" deleted

$ kubectl delete deployments --all -n astrads-system
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-deployment-support" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted

$ kubectl delete -f ../../firetap/sds/manifests/netappsdsoperator.yaml
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
```

```
customresourcedefinition.apiextensions.k8s.io
"astradscLOUDsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscLusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsdeployments.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslICENSES.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsOPTIONS.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnODEinfoES.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqOSPolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMEfiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMES.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMESnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscLOUDsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscLOUDsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscLuster-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscLuster-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslICENSE-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslICENSE-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvOLUME-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvOLUME-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
```

```

clusterrole.rbac.authorization.k8s.io "astrads-metrics-reader" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-proxy-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-proxy-rolebinding"
deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-fluent-bit-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
service "astrads-operator-metrics-service" deleted
Error from server (NotFound): error when deleting
"/.../export/firetap/sds/manifests/netappsdsoperator.yaml":
deployments.apps "astrads-operator" not found

$ kubectl get ns | grep astrads-system

[root@sti-rx2540-535c ~]# runc list
ID          PID          STATUS      BUNDLE      CREATED      OWNER

```

Astra Data Store 제거 프로세스 문제를 해결합니다

제거 프로세스 문제를 해결해야 하는 경우 다음 권장 사항을 검토하십시오.

종료 상태의 포드

Astra Data Store 제거 프로세스를 사용하면 Kubernetes에서 POD가 종료 상태로 유지되는 경우가 있습니다.

이 문제가 발생하면 다음 명령을 실행하여 "astrads-system" 네임스페이스의 모든 Pod를 강제로 삭제합니다.

```
kubectl delete pods --all -n astrads-system --force --grace-period 0
```

서비스 품질 정책은 기존 클러스터를 가리킵니다

Astra Data Store Cluster만 삭제하고 다시 배포하는 경우, QoS(Quality of Service) 정책이 이전 클러스터를 가리키므로 영구 볼륨 클레임(PVC) 또는 볼륨을 생성하지 못할 수 있습니다.

1. 이를 방지하려면 Astra Data Store 클러스터를 삭제한 후 QoS 정책을 수동으로 삭제하십시오.

```
kubectl delete AstraDSQosPolicy --all -A
```

2. 전체 Astra Data Store 구축(클러스터뿐 아니라)을 삭제합니다.

```
uninstall.sh all
```

Astra Data Store를 삭제 또는 제거한 후 키 제공자 **CRS**가 제거되지 않았습니다

삭제 또는 제거 중인 Astra Data Store 클러스터에 대해 외부 키 공급자가 구성된 경우 제거되지 않은 키 공급자 CR을 수동으로 정리해야 할 수 있습니다.

예 2. 세부 정보

다음 해결 방법을 사용합니다.

단계

1. 키 제공자 CRS가 제거되지 않았는지 확인합니다.

```
kubectl get astradskeyprovider --selector  
astrads.netapp.io/cluster=astrads-cluster-example -n astrads-system
```

응답:

NAME	AGE
externalkeyprovider1	94s

2. 키 공급자 CRS를 제거합니다.

- a. 종료자를 제거합니다.

```
kubectl edit astradskeyprovider -n astrads-system
```

- b. 아래에 강조 표시된 종료자 라인을 제거합니다.

```
kubectl edit astradskeyprovider externalkeyprovider1 -n astrads-  
system
```

```

apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSKeyProvider
metadata:
  creationTimestamp: "2022-05-24T16:38:27Z"
  finalizers:
  - astrads.netapp.io/astradskeyprovider-finalizer
  generation: 1
  labels:
    astrads.netapp.io/cluster: astrads-cluster-example
    astrads.netapp.io/rsid: "1"
  name: externalkeyprovider1
  namespace: astrads-system
  resourceVersion: "1134699"
  uid: a11111b2-31c0-4575-b7f3-97f9ab1a1bla
spec:
  cluster: astrads-cluster-example
  kmipServer:
    hostnames:
    - 10.xxx.xxx.xxx
    port: 5696
    secretRef: externalkeyprovider1
status:
  keyProviderUUID: a1b2cd34-4fc6-5bae-9184-2288c673181d
  kmipServerStatus:
    capabilities: '{ KMIP_library_version()=17367809,
KMIP_library_version_str()="KMIP
1.9.3a 8-Apr-2019", KMIP_library_version_tag()="KMIP part
of KMIP 1.9.3a 8-Apr-2019",
KMIP_library_is_eval()=false,
KMIP_library_fips_capable()=true(FIPS140),
KMIP_SSL_provider_build_version()=268444095,
KMIP_SSL_provider_version()=268444095,
KMIP_SSL_provider_version_str()="OpenSSL
1.0.2zb-fips 23 Sep 2021" }'
  keyServerUUID: 8422bdd0-74ad-579d-81bd-6d544ac4224a

```

c. 종료자를 제거한 후 키 공급자 CR을 삭제합니다.

```

kubectl delete astradskeyprovider <key-provider-cr-name> -n
astrads-system

```

VMware와 함께 Astra Data Store를 사용하십시오

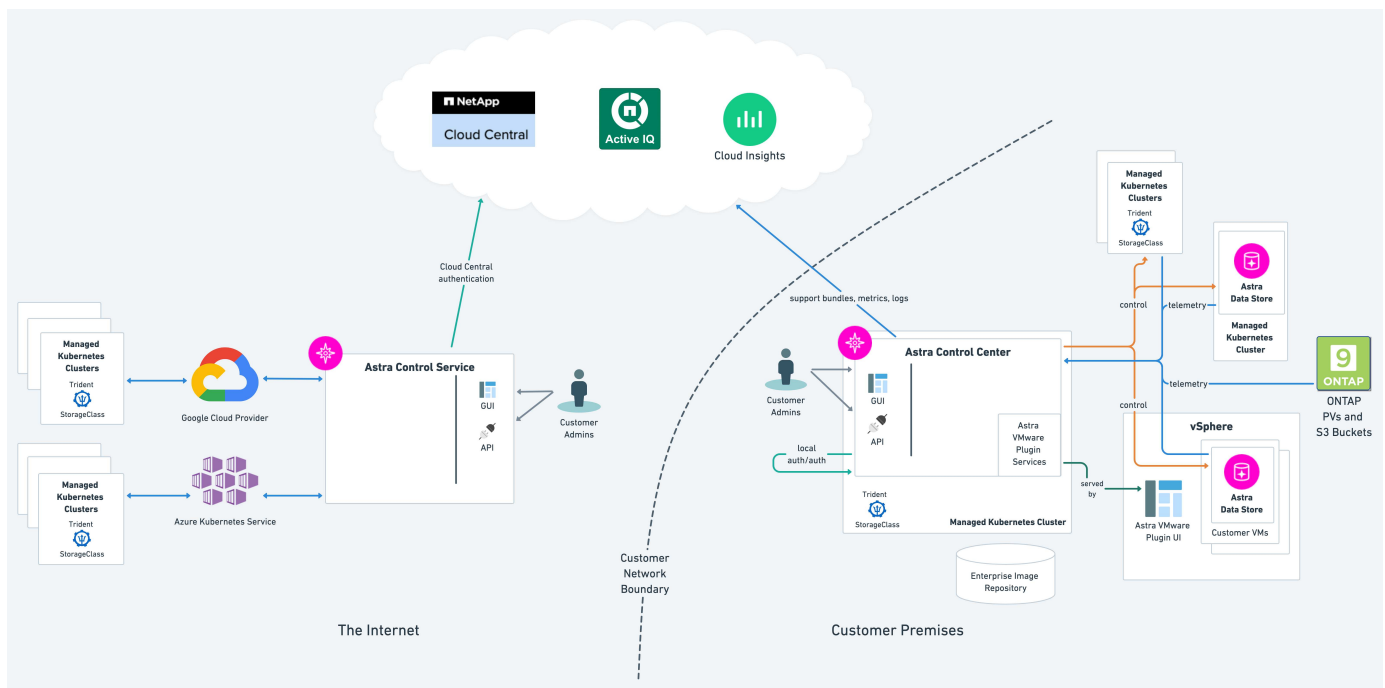
VMware를 사용한 Astra Data Store에 대해 자세히 알아보십시오

Astra Data Store는 컨테이너 워크로드와 가상화 워크로드를 모두 지원합니다. vSphere 관리자는 VVOL 및 스토리지 정책 기반 관리와 통합하여 스토리지 QoS를 적용할 수 있습니다. NetApp Astra Plugin for VMware vSphere는 익숙한 관리 및 모니터링 환경을 제공하므로 번거로운 스토리지 작업을 제거할 수 있습니다.

VMware vSphere용 Astra 플러그인은 다음과 같은 이점을 제공합니다.

- 전체 VVol 및 VASA 통합을 통한 VM 세부 스토리지 프로비저닝
- 스토리지 정책 기반 관리 통합
- vSphere 기본 관리용 vCenter 플러그인

다음 이미지는 VMware와 Astra 제품군 에코시스템을 보여 줍니다



를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)

VMware 요구 사항이 있는 Astra Data Store

사용자 환경이 일반을 충족하는지 확인한 후 ["Astra 데이터 저장소 요구 사항"](#) 또한 VMware VASA 공급자, NetApp Astra Plugin for VMware vSphere와 같은 VMware 구성 요소에 대한 최소 요구 사항도 충족하는지 확인해야 합니다.

VMware vSphere 요구 사항

Astra Data Store는 VMware VASA Provider를 API 인터페이스로 사용하여 스토리지와 통신합니다. 환경이 기본 사항을 충족하는지 확인하십시오 ["VASA 공급자 요구 사항"](#) 또한 다음과 같은 추가 요구 사항이 있습니다.

- VMware vSphere 7.0(업데이트 1부터 업데이트 3까지 지원)
- 수신 트래픽에 대해 할당되지 않은 IP 주소 1개



- NetApp Astra Plugin for VMware vSphere는 vSphere 7.0 Update 3c를 지원하지 않습니다. 대신 vSphere 7.0 Update 3D를 사용하십시오.
- NetApp Astra Plugin for VMware vSphere는 Linked Mode vCenter Server를 지원하지 않습니다.

VMware vSphere용 NetApp Astra Plugin 요구 사항

VMware vSphere용 NetApp Astra Plugin의 요구 사항은 다음과 같습니다.

- Kubernetes 클러스터에서 실행 중인 Astra Control Center 인스턴스
- Kubernetes 클러스터에서 실행 중인 라이선스 보유 Astra Data Store 인스턴스

지원되는 웹 브라우저

NetApp Astra Plugin for VMware vSphere는 다음 웹 브라우저의 최신 버전을 지원합니다.

- Mozilla Firefox
- Microsoft Edge(크롬 기반)
- Google Chrome

를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)
- ["Astra 데이터 저장소 요구 사항"](#)

VMware와 함께 Astra Data Store를 설정합니다

Astra Data Store를 스토리지 백엔드로 설정하고 VMware vSphere용 NetApp Astra Plugin을 사용하여 관리할 수 있습니다.

VMware를 사용하여 Astra Data Store를 설정하는 작업은 다음과 같습니다.

- [\[Activate VMware vSphere services using Astra Control Center\]](#).
- [\[Add a vCenter using Astra Control Center\]](#).
- [\[Create a custom SCC \(if using OpenShift\)\]](#)
- [\[Use an existing storage backend in the Astra Plugin for VMware vSphere\]](#).

- [\[Create a datastore using the Astra Plugin for VMware vSphere\]](#).
- [\[Generate VM storage policies\]](#).

VMware와 함께 Astra Data Store를 설치하기 전에 다음 사항을 확인해야 합니다.

- Astra Control Center는 **"설치되어 있습니다"** 및 설정.



Astra Data Store Early Access Program(EAP) 릴리스의 경우 Astra Control Center를 사용하여 Astra Data Store를 관리하고 VMware 워크플로우를 활성화하려면 "pcloud" 네임스페이스에만 Astra Control Center를 배포하십시오.

- Astra Data Store는 라이선스가 부여되고 배포되었습니다. 을 참조하십시오 **"Astra Data Store를 설치합니다"**.



Astra Data Store Early Access Program(EAP) 릴리스의 경우 Astra Control Center 및 Astra Data Store를 두 개의 서로 다른 Kubernetes 클러스터에 구축해야 합니다.

- Astra Control Center 및 Astra Data Store를 구축하는 데 사용된 Kubernetes 클러스터는 Astra Control Center에서 이미 관리해야 합니다.
- vCenter를 추가하기 전에 Astra Control Center 및 VASA 공급자 패키지를 업로드했습니다. **"소프트웨어 패키지 관리 를 참조하십시오"**.

Astra Control Center를 사용하여 VMware vSphere 서비스를 활성화합니다

Astra Control Center에서 vSphere 서비스를 활성화하여 VMware와 함께 Astra Data Store를 설정합니다.



Astra Control Center의 VMware vSphere 서비스는 기본적으로 설정되어 있지 않습니다.

1. Astra Control Center에 로그인합니다.
2. 왼쪽 탐색 창에서 * 클러스터 * 를 선택합니다.

배너에는 VMware vSphere 서비스가 아직 설정되지 않았다는 메시지가 표시됩니다.

3. VMware vSphere 서비스 활성화 * 를 선택합니다.

이 작업은 다소 시간이 걸릴 수 있습니다. 서비스가 활성화되면 * vCenter 추가 * 버튼이 활성화됩니다.

Astra Control Center를 사용하여 vCenter를 추가합니다

VMware vSphere용 Astra Plugin을 등록하는 첫 번째 vCenter를 추가합니다.

vCenter를 Astra Control Center에 추가하려면 관리 권한이 있어야 합니다.



VMware vSphere에 플러그인을 등록하면 VMware vSphere용 Astra Plugin 아이콘이 VMware 바로 가기 페이지에 나타납니다. VMware vSphere용 Astra Plugin을 등록해도 플러그인이 즉시 나타나지 않는 경우가 있습니다. 이 경우 몇 초 정도 기다린 후 브라우저를 새로 고치십시오.

1. Astra Control Center에 로그인합니다.

2. 왼쪽 탐색 창에서 * 클러스터 * 를 선택합니다.
3. Add vCenter * 를 선택합니다.
4. vCenter Server 세부 정보, vCenter 포트, 관리 사용자 이름 및 암호를 입력하여 Astra Control Center에 제공합니다.



이를 통해 VMware vSphere Client에 이 vCenter용 Astra Plugin을 구축할 수 있습니다.

5. 추가 * 를 선택합니다.

클러스터 페이지에 vCenter가 나타나고 관리 대상 vCenter의 총 수가 Astra Control Dashboard에 업데이트됩니다. 그러면 VMware vSphere용 Astra Plugin 구축도 시작됩니다.

vCenter 추가를 확인합니다

새로 추가된 vCenter가 클러스터 페이지와 대시보드에 나타납니다.



vCenter와 Kubernetes 클러스터가 모두 Astra Control Center 대시보드에 나타납니다.

1. Astra Control Center에 액세스합니다.
2. 왼쪽 탐색 창에서 * 클러스터 * 를 선택합니다.
3. 새로 관리되는 vCenter가 클러스터 페이지에 나타나는지 확인합니다.
4. 왼쪽 탐색 모음에서 * 대시보드 * 를 선택합니다.
5. Astra Control Center Dashboard에서 새로운 관리 vCenter 클러스터를 * Managed * 카운트의 일부로 기록합니다.



관리 클러스터 수에는 vCenter 및 Kubernetes 클러스터가 모두 포함됩니다.

6. 클러스터 세부 정보를 보려면 * Managed * 개수를 클릭합니다.

클러스터 페이지가 나타납니다.

사용자 지정 SCC 생성(OpenShift를 사용하는 경우)

OpenShift를 사용하는 경우 POD에서 수행할 수 있는 작업을 제어하고 POD에서 액세스할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 선택적으로 할당할 수 있습니다.

기본적으로 모든 컨테이너의 실행에는 제한된 SCC와 해당 SCC에 의해 정의된 기능만 부여됩니다. 제한된 SCC는 VASA 공급자 포드에 필요한 사용 권한을 제공하지 않습니다. 이 절차를 사용하여 필요한 높은 권한(샘플에 나열됨)을 VASA 공급자 구축에 사용되는 서비스 계정에 제공합니다.

권한이 부여된 노드-수출자 SCC의 혼합인 Astra Data Store 'NTV-system' 네임스페이스의 다양한 기본 서비스 계정에 사용자 지정 SCC를 할당합니다.

다음 단계는 Red Hat OpenShift Container Platform(OCP)에 배포용으로만 필요합니다.

1. Vp_backend_privileged_SCC.YAML이라는 사용자 정의 SCC를 생성한다.

```
kubectl create -f vp_backend_privileged_scc.yaml
```

샘플: vp_backend_privileged_csC.yAML

```
allowHostDirVolumePlugin: true
allowHostIPC: false
allowHostNetwork: true
allowHostPID: false
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
  - '*'
allowedUnsafeSysctls:
  - '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  name: vpbackend-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
  - '*'
supplementalGroups:
  type: RunAsAny
users:
  - system:serviceaccount:ntv-system:default
  - system:serviceaccount:ntv-system:ntv-auth-svc
  - system:serviceaccount:ntv-system:ntv-autosupport
  - system:serviceaccount:ntv-system:ntv-compliance-svc
  - system:serviceaccount:ntv-system:ntv-datastore-svc
  - system:serviceaccount:ntv-system:ntv-metallb-controller
  - system:serviceaccount:ntv-system:ntv-metallb-speaker
  - system:serviceaccount:ntv-system:ntv-mongodb
  - system:serviceaccount:ntv-system:ntv-nfs-svc
```



```

- system:serviceaccount:ntv-system:ntv-rabbitmq-svc
- system:serviceaccount:ntv-system:ntv-storage-svc
- system:serviceaccount:ntv-system:ntv-vault
- system:serviceaccount:ntv-system:ntv-vault-admin
- system:serviceaccount:ntv-system:ntv-vault-agent-injector
- system:serviceaccount:ntv-system:ntv-vault-controller
- system:serviceaccount:ntv-system:ntv-vault-initializer
- system:serviceaccount:ntv-system:ntv-vcenter-svc
- system:serviceaccount:ntv-system:ntv-vm-management-svc
- system:serviceaccount:ntv-system:ntv-watcher-svc
- system:serviceaccount:ntv-system:ntv-vault-sa-vault-tls
- system:serviceaccount:ntv-system:ntv-gateway-svc
- system:serviceaccount:ntv-system:ntv-jobmanager-svc
- system:serviceaccount:ntv-system:ntv-vasa-svc
volumes:
- '*'

```

2. OC Get SCC 명령을 사용하여 새로 추가한 SCC를 출력한다.

```
oc get scc vpbakend-privileged
```

응답:

NAME	PRIV	CAPS	SELINUX	RUNASUSER	FSGROUP	SUPGROUP
PRIORITY	READONLYROOTFS	VOLUMES				
vpbakend-privileged	true	["*"]	RunAsAny	RunAsAny	RunAsAny	RunAsAny
<no value>	false	["*"]				

VMware vSphere용 Astra Plugin에서 기존 스토리지 백엔드를 사용합니다

Astra Control Center UI를 사용하여 vCenter를 추가한 후 Astra Plugin for VMware vSphere를 사용하여 Astra Data Store 스토리지 백엔드를 추가합니다.

이 프로세스는 다음 작업을 완료합니다.

- 선택한 vCenter에 기존 스토리지 백엔드를 추가합니다.
- 선택한 vCenter에 VASA 공급자를 등록합니다. VASA Provider는 VMware와 Astra Data Store 간의 통신을 제공합니다.
- 스토리지 백엔드에 VASA 공급자 자체 서명 인증서를 추가합니다.



추가한 vCenter가 스토리지 백엔드 마법사에 나타나는 데 10분이 걸릴 수 있습니다.



Astra Data Store는 여러 vCenter와 공유해서는 안 됩니다.

단계

1. VMware vSphere용 NetApp Astra Plugin에 액세스합니다.
2. 왼쪽 탐색 창에서 * VMware vSphere * 용 Astra Plugin * 을 선택하거나 바로 가기 페이지에서 * VMware vSphere * 용 Astra Plugin 아이콘을 선택합니다.
3. VMware vSphere용 Astra Plugin 개요 페이지에서 * 기존 스토리지 백엔드 사용 * 을 선택합니다. 또는 왼쪽 탐색 창에서 * 스토리지 백엔드 * > * 추가 * 를 선택하고 * 기존 스토리지 백엔드 사용 * 을 선택합니다.
4. 스토리지 백엔드로 기존 Astra Data Store를 선택하고 * Next * 를 선택합니다.
5. VASA 공급자 페이지에서 VASA 공급자 이름, IP 주소(로드 밸런싱 장치를 사용하는 경우), 사용자 이름 및 암호를 입력합니다.



사용자 이름에는 영숫자 및 밑줄을 사용할 수 있습니다. 특수 문자를 입력하지 마십시오. 사용자 이름의 첫 문자는 영문자로 시작해야 합니다.

6. 로드 밸런싱 장치를 구축할지 여부를 표시하고 VASA 공급자에 액세스하는 데 사용할 IP 주소를 입력합니다. IP는 노드 IP와 별도로 사용 가능한 추가 IP가 되어야 합니다. 로드 밸런서가 활성화되면 Metallb는 Astra Data Store Kubernetes 클러스터에 구축되며 사용 가능한 IP를 할당하도록 구성됩니다.



Google Anthos 클러스터를 배포용으로 사용하는 경우, Anthos가 이미 로드 밸런서로 메타 디바이스를 실행하므로 로드 밸런서를 배포하지 않도록 선택합니다. VASA 공급자 CR(v1beta1_vasaprovider.YAML)에서는 메타 배포 플래그를 false로 설정해야 합니다.

로드 밸런서를 배포하지 않도록 선택한 경우 로드 밸런서가 이미 구축되어 있으며 * 로드 밸런서 * 유형의 Kubernetes 서비스에 대한 IP를 할당하도록 구성되어 있다고 가정합니다.



이 구축 시점에는 VASA Provider가 아직 구축되지 않았습니다.

7. 다음 * 을 선택합니다.
8. 인증서 페이지에서 자체 서명된 인증서의 인증서 정보를 검토합니다.
9. 다음 * 을 선택합니다.
10. 요약 정보를 검토합니다.
11. 추가 * 를 선택합니다.

이렇게 하면 VASA Provider가 구축됩니다.

VMware vSphere용 Astra Plugin에서 스토리지 백엔드를 확인합니다

Astra Data Store 스토리지 백엔드가 등록되면 VMware vSphere용 Astra Plugin 스토리지 백엔드 목록에 나타납니다.

스토리지 백엔드 상태와 VASA 공급자 상태를 확인할 수 있습니다. 각 스토리지 백엔드의 사용된 용량도 확인할 수 있습니다.

스토리지 백엔드를 선택한 후 사용된 용량과 사용 가능한 용량, 데이터 축소율 및 내부 네트워크 관리 IP 주소를 볼 수도 있습니다.

단계

1. NetApp Astra Plugin for VMware vSphere의 왼쪽 탐색 모음에서 * Storage backends * 를 선택합니다.

2. 요약 탭을 보려면 Astra Data Store 스토리지 백엔드를 선택합니다.
3. VASA Provider의 Used 및 Available Capacity, Data Reduction Ratio 및 Status를 검토합니다.
4. 다른 탭을 선택하여 VM, 데이터 저장소, 호스트 및 스토리지 노드에 대한 정보를 확인합니다.

VMware vSphere용 Astra Plugin을 사용하여 데이터 저장소를 생성합니다

스토리지 백엔드를 추가하고 VMware vSphere용 Astra Plugin을 등록하면 VMware에서 데이터 저장소를 생성할 수 있습니다.

데이터 센터, 컴퓨팅 또는 호스트 클러스터에 데이터 저장소를 추가할 수 있습니다.



동일한 스토리지 백엔드를 사용하여 동일한 데이터 센터에 여러 데이터 저장소를 생성할 수 없습니다.

NFS 프로토콜을 사용하여 VVOL 데이터 저장소 유형을 추가할 수 있습니다.

단계

1. VMware vSphere용 Astra Plugin에 액세스합니다.
2. 플러그인 메뉴에서 * Create Datastore * 를 선택합니다.
3. 새 데이터 저장소 이름, 유형(VVol) 및 프로토콜(NFS)을 입력합니다.
4. 다음 * 을 선택합니다.
5. 스토리지 페이지에서 방금 생성한 Astra Data Store 스토리지 백엔드를 선택합니다.



기존 데이터 저장소가 있는 스토리지 백엔드는 사용할 수 없습니다.

6. 다음 * 을 선택합니다.
7. 요약 페이지에서 정보를 검토합니다.
8. Create * 를 선택합니다.



스캔 실패 또는 일반 시스템 오류와 관련된 오류가 발생하면 "vCenter에서 스토리지 공급자를 다시 검색/동기화합니다" 그런 다음 데이터 저장소를 다시 생성하십시오.

VM 스토리지 정책을 생성합니다

데이터 저장소를 생성한 후 VM을 생성하기 전에 REST API UI에서 '/virtualization/api/v1/vCenters/vm-storage-policies'를 사용하여 미리 디자인된 VM 스토리지 정책을 생성해야 합니다.

단계

1. REST API UI 페이지는 https://<ads_gateway_ip>:8443 으로 이동합니다.
2. API 'POST/virtualization/API/auth/login'으로 이동하여 사용자 이름, 암호 및 vCenter 호스트 이름을 입력합니다.

응답:

```
{
  "vmware-api-session-id": "212f4d6447b05586ab1509a76c6e7da56d29cc5b",
  "vcenter-guid": "8e475060-b3c8-4267-bf0f-9d472d592d39"
}
```

3. API 'get/virtualization/api/auth/validate-session'으로 이동하여 다음 단계를 완료합니다.

- 위에서 생성한 VMware-API-session-id와 vCenter-GUID를 헤더로 사용합니다.
- 지금 체험하기 * 를 선택합니다.

응답: (아래에서 인증이 잘림):

```
authorization: eyJhbGciOiJSUzI1NiIsInR...9h15DYYvClT3oA connection:
keep-alive content-type: application/json date: Wed, 18 May 2022
13:31:18 GMT server: nginx transfer-encoding: chunked
```

4. API 'virtualization/api/v1/vCenters/vmVM-storage-policies'로 이동하여 이전 응답에서 생성된 베어러 토큰을 'authorization'으로 추가합니다.

"200" 응답이 나타나고 세 개의 VM 스토리지 정책이 생성됩니다.

- vCenter 스토리지 정책 페이지에서 새 VM 스토리지 정책(Bronze, Silver, Gold 이름)을 확인합니다.
- VM을 생성하여 계속합니다.

다음 단계

다음 작업을 수행할 수 있습니다.

- VM을 생성합니다.
- 데이터 저장소를 마운트합니다. 을 참조하십시오 ["데이터 저장소를 마운트합니다"](#).

를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)

VMware 설치의 구성 요소를 모니터링합니다

NetApp Astra Plugin for VMware vSphere를 사용하여 Astra Data Store 설치의 구성 요소를 모니터링할 수 있습니다. 스토리지 백엔드, VASA 공급자, VM 및 VVOL을 비롯한 시스템의 상태를 모니터링할 수 있습니다. 용량 및 vCenter 정보를 볼 수도 있습니다.

VMware vSphere용 Astra Plugin 대시보드를 사용하여 시스템 상태를 모니터링합니다

VMware 환경에서 Astra Data Store를 관리하려면 스토리지 백엔드 및 VASA 공급업체의 전반적인 상태를 모니터링해야 합니다.

NetApp Astra Plugin for VMware vSphere 대시보드를 사용하여 다음 정보를 확인할 수 있습니다.

- 이 vCenter에서 사용되는 모든 스토리지 백엔드의 물리적 용량 및 사용 가능한 용량입니다. 정보 위로 마우스를 가져가면 자세한 정보를 볼 수 있습니다.
- 정상 상태가 아닌 스토리지 백엔드 및 VASA 공급자
- 상위 10개 VM의 지연 시간, IOPS, 처리량, 용량 활용률

대시보드에서 다음과 같은 몇 가지 추가 작업을 수행할 수 있습니다.

- 용량을 모니터링합니다
- 기존 스토리지 백엔드를 사용합니다. 을 참조하십시오 ["저장소 백엔드를 설정합니다"](#).
- 제품 설명서에 액세스합니다

대시보드 검토 단계

1. VMware vSphere용 Astra Plugin에 액세스합니다.
2. 개요 페이지에서 다음 섹션을 검토합니다.
 - a. * 스토리지 백엔드 * 섹션: 스토리지 백엔드와 VASA 공급자 모두의 상태를 클릭하여 해당 상태에 대한 세부 정보를 볼 수 있습니다. 을 클릭하여 모든 스토리지 백엔드를 볼 수도 있습니다.
 - b. * 스토리지 백엔드 용량 * 섹션: 선택한 vCenter에서 스토리지 백엔드에 대해 사용된 총 물리적 용량과 사용 가능한 용량을 검토합니다. vCenter Server를 변경하려면 오른쪽 위에 있는 vCenter Server 옵션을 클릭합니다.
 - c. * 가상 머신 * 섹션: 10대 용량 사용률이 가장 높은 가상 머신을 검토합니다.



대신 표 제목을 클릭하여 대기 시간이 긴 상위 10개 VM과 같이 원하는 항목을 표시할 수 있습니다.

다른 보기에서 **Astra Data Store**를 모니터링하는 단계입니다

1. Astra Data Store 구성 요소를 모니터링하려면 다음 보기에 액세스하십시오.
 - * 가상 머신 * 탭: 최상위 10개의 VM만 나열되는 대시보드와 비교하여 Astra Data Store에서 관리하는 모든 VM을 나열합니다.
 - * Storage * 드릴다운: 스토리지 시스템과 연결된 호스트, 가상 머신 및 데이터 저장소를 표시합니다.
 - * VM Storage * view(VM 스토리지 * 보기): VASA 공급자가 생성한 VVOL에 대한 세부 정보를 제공합니다.

스토리지 백엔드 임계값 설정을 검토합니다

스토리지 백엔드 용량 임계값 설정은 스토리지 백엔드의 모든 데이터 저장소에 알림 메시지가 표시되는 시기를 제어합니다.

Astra Plugin for VMware vSphere를 사용하여 스토리지 백엔드를 구축 또는 추가할 때 다음과 같은 기본 임계값이 설정됩니다.

- 90% 차 있는 경우 빨간색 경고가 생성됩니다
- 80% 차 있음 은 노란색 경고를 생성합니다

VMware에서 시스템에서 알림을 생성하는 레벨을 볼 수 있습니다.



Astra Data Store Early Access Program의 경우 여러 데이터 센터에서 동일한 스토리지 컨테이너를 사용하는 경우 데이터 저장소에 대한 잘못된 알람이 표시될 수 있습니다.

단계

1. VMware vSphere용 NetApp Astra Plugin에 액세스합니다.
2. 왼쪽 탐색 창에서 * 설정 * 을 선택합니다.
3. 설정 임계값을 검토합니다.

를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)

VMware 설치의 Astra Data Store 구성 요소를 관리합니다

vSphere 환경과 Astra Control Center에서 다음과 같은 Astra Data Store 구성 요소를 관리할 수 있습니다.

- [\[Work with managed vCenters\]](#)
- [\[Manage VMs from vSphere\]](#)
- [\[Manage the storage backend\]](#)
- [\[Manage datastores\]](#)

관리되는 **vCenter**와 함께 작업합니다

다음과 같은 방법으로 관리되는 vCenter를 사용할 수 있습니다.

- [\[View vCenter details in Astra Control Center\]](#)
- [\[View vCenter details in Astra Control Center\]](#)
- [\[Unmanage a vCenter in Astra Control Center\]](#)

Astra Control Center에서 **vCenter** 세부 정보를 봅니다

클러스터와 연결된 모든 vCenter를 확인할 수 있습니다.

단계

1. Astra Control Center 왼쪽 탐색 창에서 * Clusters * 를 선택합니다.
2. vCenter 목록을 봅니다.
3. 저장 * 을 선택합니다.

Astra Control Center에서 vCenter 세부 정보를 봅니다

시스템 및 클러스터의 상태를 볼 수 있습니다. Astra Control Center 대시보드를 보면 몇 개의 클러스터를 관리할 수 있는지 확인할 수 있습니다.

단계

1. Astra Control Center 왼쪽 탐색 창에서 * Clusters * 를 선택합니다.
2. vCenter를 선택합니다.
3. 정보를 봅니다.

Astra Control Center에서 vCenter 관리를 해제합니다

Astra Control Center에서 vCenter를 더 이상 관리하지 않으려면 관리를 해제할 수 있습니다. 그러면 Astra Control Center에서 vCenter가 제거되고 등록이 해제됩니다.



먼저 이 vCenter와 연결된 클러스터, 스토리지 백엔드 및 VM을 VMware vSphere용 Astra 플러그인에서 제거해야 합니다.

단계

1. Astra Control Center 왼쪽 탐색 창에서 * Clusters * 를 선택합니다.
2. 클러스터 페이지에서 vCenter를 선택합니다.



또는 여러 vCenter를 선택하고 * Unmanage All * 을 선택합니다.

3. Actions * 메뉴를 마우스 오른쪽 버튼으로 클릭하고 * Unmanage * 를 선택합니다.
4. vCenter 관리 해제 페이지에서 "unmanage"를 입력합니다.
5. Yes, unmanage vcenter * 를 선택합니다.

vSphere에서 VM을 관리합니다

기본 vSphere 작업을 사용하여 Astra Data Store와 연결된 VM을 관리할 수 있습니다.

- "VM을 삭제합니다"
- "VM의 이름을 바꿉니다"
- "VM 크기를 조정합니다"



이 릴리즈에서는 한 번에 하나의 VM 디스크만 크기를 조정할 수 있습니다. 여러 디스크의 크기를 조정하려고 하면 실패합니다.

- "VM의 전원을 켜거나 끕니다"
- "VM을 일시 중단합니다"
- "VM을 재설정합니다"

기본 vCenter 작업을 사용하는 Astra Data Store에 대해 다음과 같은 스냅샷 워크플로우를 사용할 수 있습니다.

- "Astra Data Store의 스냅샷을 생성합니다"
- "스냅샷을 복구합니다"
- "스냅샷을 삭제합니다"



VVOL 런타임 오류로 인해 스냅샷 작업이 실패하는 경우가 있습니다. 이 경우 작업을 다시 시도하십시오.

스토리지 백엔드를 관리합니다

스토리지 백엔드를 제거할 수 있습니다. 스토리지 백엔드를 제거해도 데이터가 삭제되는 것이 아니라 Astra Data Store 제품 자체를 삭제하지 않습니다. VMware에서 VASA 공급자를 등록 해제하고 해당 vCenter에 대한 스토리지 백엔드의 링크를 해제하기만 하면 됩니다.



VASA Provider가 vCenter 외부에서 설정 및 구축된 경우 Astra Data Store만 제거할 수 있습니다. 스토리지 백엔드가 데이터 저장소 프로비저닝 프로세스의 일부로 사용되는 경우 스토리지 백엔드를 제거할 수 없습니다.

Astra Data Store가 둘 이상의 vCenter와 연결되어 있지 않은 경우 vCenter를 제거하면 VASA Provider가 등록 취소되고 제거됩니다.

단계

1. VMware vSphere용 Astra Plugin에 액세스합니다.
2. 왼쪽 탐색 창에서 * 스토리지 백엔드 * 를 선택합니다.
3. 스토리지 백엔드 페이지에서 스토리지 백엔드 작업 메뉴를 클릭하고 * 제거 * 를 선택합니다.
4. VASA 공급자 사용자 이름 및 암호를 입력합니다.
5. 제거 * 를 선택합니다.

데이터 저장소를 관리합니다

기본 vCenter 작업을 사용하여 vSphere 환경에서 Astra Data Store를 관리하고, VM 및 Astra Plugin 확장을 통해 데이터 저장소를 관리할 수 있습니다.

- "데이터 저장소를 생성합니다"
- [\[Mount a datastore\]](#)
- [\[Delete a datastore\]](#)

데이터 저장소를 마운트합니다

VMware vSphere용 Astra 플러그인을 사용하여 하나 이상의 추가 호스트에 데이터 저장소를 마운트할 수 있습니다.

단계

1. vCenter의 데이터 센터 인벤토리에서 Astra Data Store용 데이터 저장소를 선택합니다.
2. 데이터 저장소를 마우스 오른쪽 버튼으로 클릭하고 * VMware vSphere * 용 Astra Plugin > * 데이터 저장소 마운트 * 를 선택합니다.
3. Mount datastore on hosts 페이지에서 데이터 저장소를 마운트할 호스트를 선택합니다.



모든 호스트에 데이터 저장소를 마운트하려면 * 모든 호스트에 마운트 * 를 선택합니다.

4. Mount * 를 선택합니다.

작업을 시작한 후 vSphere Client의 Recent Tasks 패널에서 진행률을 추적할 수 있습니다.



스캔 실패 또는 일반 시스템 오류와 관련된 오류가 발생하면 ["vCenter에서 스토리지 공급자를 다시 검색/동기화합니다"](#) 그런 다음 데이터 저장소를 다시 생성하십시오.

데이터 저장소를 삭제합니다

VMware vSphere용 Astra 플러그인을 사용하여 데이터 저장소를 삭제할 수 있습니다.



데이터 저장소를 삭제하려면 먼저 데이터 저장소의 모든 VM을 제거해야 합니다.

단계

1. vCenter의 데이터 센터 인벤토리에서 데이터 저장소를 선택합니다.
2. 데이터 저장소를 마우스 오른쪽 버튼으로 클릭하고 * Astra Plugin * > * Delete Datastore * 를 선택합니다.
3. 데이터 저장소 삭제 페이지에서 정보를 확인하거나 추가 권장 조치를 수행하여 데이터 저장소를 삭제할 수 있습니다.
4. 삭제 * 를 선택합니다.

를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)

VMware 통합 환경에서 Astra Data Store를 제거합니다

vSphere 환경에서 Astra Data Store 및 관련 구성 요소를 제거할 수 있습니다.

을 참조하십시오 ["참조하십시오"](#) Astra Data Store를 제거하는 중입니다.

를 참조하십시오

- ["Astra Control Center 문서"](#)
- ["Astra 제품군 소개"](#)

지식 및 지원

문제 해결

발생할 수 있는 몇 가지 일반적인 문제를 해결하는 방법에 대해 알아봅니다.

https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra

도움을 받으십시오

NetApp은 다양한 방법으로 Astra Data Store를 지원합니다. 무료 셀프 서비스 지원 옵션은 기술 자료(KB) 기사 및 Slack 채널과 같이 24x7 이용할 수 있습니다.



Astra Data Store에 대한 커뮤니티 기술 지원을 받을 수 있습니다. 를 사용하여 케이스를 생성합니다 "NetApp Support 사이트(NSS)" 은(는) 이 릴리스에서 사용할 수 없습니다. 피드백 옵션을 통해 지원 팀에 문의하거나 Slack 채널을 사용하여 셀프 서비스를 받을 수 있습니다.

셀프 서비스 지원 옵션

이러한 옵션은 24x7 무료로 제공됩니다.

- "기술 자료(로그인 필요)"

Astra Data Store와 관련된 문서, FAQ 또는 고장 수리 정보를 검색합니다.

- 문서화

현재 보고 있는 문서 사이트입니다.

- "NetApp "containers" Slack 채널"

"containers" 채널로 이동하여 동료 및 전문가와 교류하십시오.

- 피드백 이메일

귀하의 생각, 아이디어 또는 우려 사항을 당사에 알릴 수 있도록 astra.feedback@netapp.com 으로 이메일을 보내주십시오.

자세한 내용을 확인하십시오

- "NetApp에 파일을 업로드하는 방법(로그인 필요)"
- "NetApp 기술 자료 문서"

자동 지원 모니터링

AutoSupport는 Astra Data Store 시스템의 런타임 및 정보를 모니터링하여 메시지를 NetApp 지원으로 보냅니다. 이러한 시스템 구성 요소는 구성에 따라 모니터링할 수 있습니다.

- 컨트롤 플레인
- 스토리지

AutoSupport는 기본적으로 이 기간 동안 사용하도록 설정됩니다. "[Astra Data Store 클러스터 설치](#)" 또는 AutoSupport CR(사용자 지정 리소스)이 클러스터에 적용된 후 활성화되면 AutoSupport(ASUP) 번들이 에 자동으로 업로드됩니다. "[NetApp Support 사이트\(NSS\)](#)" 또는 수동 다운로드에 사용할 수 있습니다.

옵션

- [\[AutoSupport triggers and scenarios\]](#)
- [\[Configure custom control plane AutoSupport collection\]](#)
- [\[Configure custom storage AutoSupport collection\]](#)
- [\[List ASUPs in the system\]](#)
- [\[Download an ASUP Bundle\]](#)
- [\[Upload a core file\]](#)

AutoSupport 트리거 및 시나리오

AutoSupport 번들은 다음과 같은 방법으로 트리거됩니다.

- * 주기적으로 *: CR에서 정의된 간격으로 ASUP 번들을 생성합니다.
- * 사용자 트리거됨 *: 로그를 볼 수 있도록 사용자 고유의 ASP를 수동으로 생성할 수 있습니다.
- * 코어 덤프 *: 노드에 코어 덤프가 있으면 ASUP가 생성되고 코어가 NetApp으로 전송되어 추가 조사가 수행됩니다.
- * Callhome 이벤트 기반 *: 운영 체제에서 특정 callhome 이벤트를 통해 ASUP가 생성됩니다.
- * Kubernetes 이벤트 기반 *: ASUP는 제어 플레인 내 특정 Kubernetes 이벤트에서 생성됩니다.

이러한 트리거 시나리오는 다음 AutoSupport 유형 중 하나를 생성합니다.

- * ControlPlane AutoSupport *: Astra 데이터 저장소 컨트롤 플레인 로그 및 CRS의 모음입니다.
- * Storage AutoSupport *: 스토리지 보고서 및 성능 데이터의 모음입니다.
- * 코어 덤프 AutoSupport *: 시스템 코어 덤프의 모음입니다.

사용자 지정 컨트롤 플레인 **AutoSupport** 컬렉션을 구성합니다

컨트롤 플레인 이벤트를 보고하는 사용자 지정 AutoSupport 컬렉션 구성을 만들 수 있습니다. 대부분의 설치의 기본적으로 에 대해 정기적인 이벤트 보고를 사용하도록 설정되어 있습니다. "[Astra Data Store 클러스터 설치](#)". 이 절차에서는 선택한 매개 변수를 기반으로 보고하는 AutoSupport CR을 구성하는 방법에 대해 설명합니다.

단계

1. 컨트롤 플레인 컬렉션 CR을 만들려면 다음 명령을 사용자 지정합니다.

```
kubectl astasds asup collect -c controlplane --namespace=astrads-system
```

- a. 사용자 정의 매개변수 정의:

- "<myASUPname>": 생성할 AutoSupport CR의 이름입니다.
- '-e<event name>': 수집을 트리거하는 이벤트 이름입니다. 이벤트 이름은 구성 요소에 미리 정의되어 있어야 합니다. YAML(지원 컨트롤러에 탑재됨).

예:

```
kubectl astrasds asup collect -c controlplane custom-asup-name -e debug
--namespace=astrads-system
```

a. 시스템에 필요한 추가 매개 변수를 추가합니다.

- '--클러스터': 이 플래그는 다중 클러스터 환경에서 필요합니다.
- '--localCollection': 로컬 수집을 활성화합니다. 기본값은 false 입니다.
- '--forceUpload': 강제 업로드가 가능합니다. 기본값은 false 입니다.
- '--RETRY': 재시도를 활성화합니다. 기본값은 false 입니다.

사용자 지정 저장소 **AutoSupport** 컬렉션을 구성합니다

스토리지 구성 요소 이벤트를 보고하는 사용자 지정 AutoSupport 컬렉션 구성을 생성할 수 있습니다. 대부분의 설치 기본적으로 에 대해 정기적인 이벤트 보고를 사용하도록 설정되어 있습니다 ["Astra Data Store 클러스터 설치"](#). 이 절차에서는 선택한 매개 변수를 기반으로 보고하는 AutoSupport CR을 구성하는 방법에 대해 설명합니다.

단계

1. 다음 명령을 사용자 지정하여 스토리지 컬렉션 CR을 생성합니다.

```
kubectl astrasds asup collect -c storage --namespace=astrads-system
```

a. 사용자 정의 매개변수 정의:

- "<myASUPname>": 생성할 AutoSupport CR의 이름입니다.
- '-e<event name>': 수집을 트리거하는 이벤트 이름입니다. 이벤트 이름은 구성 요소에 미리 정의되어 있어야 합니다. YAML(지원 컨트롤러에 탑재됨).

성능 이벤트의 예:

```
kubectl astrasds asup collect -c storage -e performance example-perf-storage-asup
```

- "-t<iso_format>-d<hours>": 지정된 기간 동안 모든 노드에 대해 스토리지 ASUP를 수집합니다. 표준 ISO 날짜 시간 형식("-t")을 시간(dA) 단위로 사용합니다. 예를 들면 다음과 같습니다.

```
kubectl astrasds asup collect -c storage -t 2021-01-01T15:00:00Z -d 24
```

- '--nodename>': 지정된 노드에 대한 스토리지 ASUP를 수집합니다. 예를 들면 다음과 같습니다.

```
kubectl astrads asup collect -c storage --nodes example1
```

- '- 노드 nodename1,nodename2,nodename3': 지정된 노드에 대한 스토리지 ASUP 수집:

```
kubectl astrads asup collect -c storage --nodes
example1,example2,example3
```

a. 시스템에 필요한 추가 매개 변수를 추가합니다.

- '--클러스터': 이 플래그는 다중 클러스터 환경에서 필요합니다.
- '-- localCollection ': 로컬 수집을 활성화합니다. 기본값은 false 입니다.
- '--forceUpload': 강제 업로드가 가능합니다. 기본값은 false 입니다.
- '--RETRY': 재시도를 활성화합니다. 기본값은 false 입니다.

시스템에 **ASP**를 나열합니다

다음 명령을 사용하여 시스템의 ASP를 이름으로 나열할 수 있습니다.

```
kubectl astrads asup list --namespace=astrads-system
```

샘플 반응:

NAMESPACE	NAME	SEQUENCE	NUMBER	EVENT
SIZE	STATE	LOCAL	COLLECTION	
astrads-system	storage-callhome.reboot.unknown-...	1		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-tdl2h:	
astrads-system	storage-callhome.reboot.unknown-...	2		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-xx6n8:	
astrads-system	storage-callhome.reboot.unknown-...	3		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-qghnx:	

ASUP 번들을 다운로드하십시오

이 명령을 사용하여 로컬에서 수집한 ASUP 번들을 다운로드할 수 있습니다. 현재 작업 디렉토리 이외의 위치를 지정하려면 '-o<location>'을 사용합니다.

```
./kubectl-astrads asup download <ASUP_bundle_name> -o <location>
```

코어 파일을 업로드합니다

서비스가 충돌하면 충돌 시 관련 메모리 콘텐츠가 포함된 파일(코어 파일이라고 함)과 함께 AutoSupport(ASUP) 메시지가 생성됩니다. Astra Data Store가 ASUP 메시지를 NetApp Support에 자동으로 업로드하지만, ASUP 메시지와 연관된 코어 파일을 수동으로 업로드해야 합니다.

단계

1. 다음 "kubctl" 명령을 사용하여 ASUP 메시지를 확인하십시오.

```
kubect1 astrasds asup list --namespace=astrads-system
```

다음과 유사한 출력이 표시됩니다.

NAMESPACE	NAME	SEQUENCE NUMBER	EVENT
SIZE	STATE	LOCAL COLLECTION	
astrads-system	storage-coredump-2021...	1	coredump
197848373	compressed	astrads-ds-support-sxxn7:/var/...	

2. 다음 "kubctl" 명령을 사용하여 ASUP 메시지에서 핵심 파일을 다운로드합니다. 다운로드한 파일의 대상 디렉토리를 지정하려면 '-o' 옵션을 사용합니다.

```
kubect1 astrads asup download storage-coredump-20211216t140851311961680  
-o <absolute_path_to_destination_directory>
```



다른 핵심 파일이 삭제되어 코어 파일을 다운로드하지 못하는 경우가 드물게 있습니다. 이 경우 "Cannot stat: No such file or directory" 오류가 반환됩니다. 이 오류가 표시되면 를 사용할 수 있습니다 ["도움을 받으십시오"](#).

3. 웹 브라우저를 열고 로 이동합니다 ["NetApp 인증된 파일 업로드 툴"](#), 아직 로그인하지 않은 경우 NetApp 지원 자격 증명을 입력합니다.
4. 케이스 번호가 없습니다 * 확인란을 선택합니다.
5. 가장 가까운 지역 * 메뉴에서 가장 가까운 지역을 선택합니다.
6. 업로드 * 버튼을 선택합니다.
7. 이전에 다운로드한 코어 파일을 찾아 선택합니다.

업로드가 시작됩니다. 업로드가 완료되면 성공 메시지가 나타납니다.

자세한 내용을 확인하십시오

- ["NetApp에 파일을 업로드하는 방법\(로그인 필요\)"](#)

이전 버전의 **Astra Data Store**

이전 릴리스에 대한 문서를 사용할 수 있습니다.

- ["Astra Data Store 21.12 문서"](#)

법적 고지

법적 고지 사항은 저작권 선언, 상표, 특허 등에 대한 액세스를 제공합니다.

저작권

<http://www.netapp.com/us/legal/copyright.aspx>

상표

NetApp, NetApp 로고, NetApp 상표 페이지에 나열된 마크는 NetApp Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.

<http://www.netapp.com/us/legal/netapptmlist.aspx>

특허

NetApp 소유 특허 목록은 다음 사이트에서 확인할 수 있습니다.

<https://www.netapp.com/us/media/patents-page.pdf>

개인 정보 보호 정책

<https://www.netapp.com/us/legal/privacypolicy/index.aspx>

오픈 소스

통지 파일은 NetApp 소프트웨어에 사용된 타사의 저작권 및 라이선스에 대한 정보를 제공합니다.

["Astra Data Store EAP 릴리스에 대한 고지 사항"](#)

저작권 정보

Copyright © 2022 NetApp, Inc. All rights reserved. 미국에서 인쇄된 본 문서의 어떤 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 그래픽, 전자적 또는 기계적 수단(사진 복사, 레코딩 등)으로도 저작권 소유자의 사전 서면 승인 없이 전자 검색 시스템에 저장 또는 저장.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지 사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 "있는 그대로" 제공되며 상품성 및 특정 목적에 대한 적합성에 대한 명시적 또는 묵시적 보증을 포함하여 이에 제한되지 않고, 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 또는 파생적 손해(소계 물품 또는 서비스의 조달, 사용 손실, 데이터 또는 수익 손실, 계약, 엄격한 책임 또는 불법 행위(과실 또는 그렇지 않은 경우)에 관계없이 어떠한 책임도 지지 않으며, 이는 이러한 손해의 가능성을 사전에 알고 있던 경우에도 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 구입의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

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

권리 제한 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.277-7103(1988년 10월) 및 FAR 52-227-19(1987년 6월)의 기술 데이터 및 컴퓨터 소프트웨어의 권리(Rights in Technical Data and Computer Software) 조항의 하위 조항 (c)(1)(ii)에 설명된 제한사항이 적용됩니다.

상표 정보

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