



# **Astra Data Store** 미리 보기 문서

## Astra Data Store

NetApp  
January 31, 2022

# 목차

Astra Data Store 미리 보기 문서	1
릴리스 정보	2
Astra Data Store 이번 릴리스의 내용	2
알려진 문제	2
알려진 제한 사항	3
개념	5
Astra 데이터 저장소 미리 보기 소개	5
Astra Data Store 구축 모델을 미리 봅니다	7
클러스터 확장	9
Astra Data Store 미리보기 내의 스토리지 효율성	9
Astra Data Store 미리 보기의 보안	10
시작하십시오	11
Astra Data Store 미리 보기 요구 사항	11
Astra 데이터 저장소 미리 보기를 빠르게 시작합니다	14
Astra Data Store 설치 개요	15
Astra Data Store 미리보기 구성 요소를 설정합니다	42
Astra Data Store 미리보기 제한	52
Astra Data Store 미리 보기를 위한 질문과 대답	52
Astra Data Store를 사용하십시오	56
kubectl 명령을 사용하여 Astra Data Store 미리 보기 자산을 관리합니다	56
테스트 응용 프로그램을 배포합니다	62
클러스터를 관리합니다	66
Cloud Insights를 통해 메트릭을 모니터링합니다	78
이벤트 로그를 구성하고 모니터링합니다	90
Astra Data Store 미리 보기와 함께 Astra Control Center를 사용합니다	91
Astra Data Store 미리 보기를 제거합니다	91
지식 및 지원	98
문제 해결	98
도움을 받으십시오	98
자동 지원 모니터링	98
법적 고지	103

# Astra Data Store 미리 보기 문서

# 릴리스 정보

Astra Data Store의 21.12 Preview 릴리스를 발표하게 되어 기쁘게 생각합니다.

- ["Astra Data Store 이번 릴리즈에는 어떤 내용이 포함되어 있는지"](#)
- ["알려진 문제"](#)
- ["알려진 제한 사항"](#)

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

## Astra Data Store 이번 릴리스의 내용

Astra Data Store의 21.12 Preview 릴리스를 발표하게 되어 기쁘게 생각합니다.

### 2021년 12월 21일(21.12)

Astra Data Store Preview 초기 릴리즈.

- ["그게 뭐죠"](#)
- ["구축 모델 및 구성 요소"](#)
- ["시작하는 데 필요한 사항"](#)
- ["설치합니다"](#) 및
- ["관리"](#) 및 성능
- ["Cloud Insights를 사용한 모니터링"](#)
- ["도움을 받으십시오"](#) 및

자세한 내용을 확인하십시오

- ["알려진 문제"](#)
- ["알려진 제한 사항"](#)
- ["NetApp 기술 자료 문서"](#)

## 알려진 문제

알려진 문제점은 제품의 이 미리보기 릴리스를 성공적으로 사용하지 못하게 할 수 있는 문제를 식별합니다.

기본 **Calico WireGuard** 인터페이스 이름은 **Astra Data Store** 미리 보기에서 인식되지 않습니다

이 오류로 인해 포드가 충돌합니다. 해결 방법으로 Calico의 Felix 구성에서 `wirreguardInterfaceName` 설정을 변경하여 점 없이 인터페이스 이름을 지정합니다. 예를 들면 다음과 같습니다.

```
calicoctl patch felixconfiguration default --type='merge' -p
'{"spec":{"wireguardInterfaceName":"calico_wg"}}'
```

새 이름을 사용하여 각 노드에 인터페이스를 생성하는 것 외에도 이 변경 사항은 원래 wireguard.cali 인터페이스 뒤에 남겨질 수도 있습니다. 이 문제가 반복되지 않도록 이 인터페이스를 삭제합니다.

**VXLAN** 구성 및 **Calico CNI**의 네트워크 인터페이스 이름은 **Astra Data Store** 미리보기에서 인식되지 않습니다

이 오류로 인해 포드가 충돌합니다. 이 문제를 해결하려면 VXLAN 구성을 비활성화하십시오.

기본 활성 프로브 값을 가진 **MongoDB** 구축은 충돌 루프에서 **Pod**와 함께 실패합니다

해결 방법으로 MongoDB 구축 사양에서 활성 프로브 initialDelaySeconds를 600초로 설정합니다.

자세한 내용을 확인하십시오

- ["알려진 제한 사항"](#)

## 알려진 제한 사항

알려진 제한 사항은 이 제품의 미리 보기 릴리스에 의해 지원되지 않거나 올바르게 상호 운용되지 않는 플랫폼, 장치 또는 기능을 식별합니다. 이러한 제한 사항을 주의 깊게 검토하십시오.

하나 이상의 노드를 제거하는 기능은 지원되지 않습니다

Astra Data Store 미리 보기는 장애가 발생한 노드 교체를 지원하지만 노드 제거 기능은 지원하지 않습니다.

드라이브를 추가하거나 제거하는 기능은 지원되지 않습니다

Astra Data Store 미리 보기는 장애가 발생한 드라이브 교체를 지원하지만 기존 클러스터에 드라이브를 추가하거나 기존 클러스터에서 드라이브를 제거하는 기능은 지원하지 않습니다.

**Astra Data Store** 미리보기 기능은 방화벽이 활성화된 상태에서 검증되지 않았습니다

Astra Data Store 미리 보기를 사용하려면 호스트 방화벽(firwalld)을 비활성화해야 합니다. Calico HostEndpoint와 같은 CNI 도구를 사용하여 방화벽을 활성화하는 것은 검증되지 않았습니다.

업그레이드 또는 패치를 설치하려면 새로 설치해야 합니다

Astra Data Store 미리 보기는 운영 워크로드를 대상으로 하지 않습니다.

**Ubuntu** 기반 베어 메탈 또는 **VM** 패스스루 배포에는 **NVMe TLC SSD**가 필요합니다

이 제한은 RHEL, RHCOS 또는 CentOS 기반 배포에는 적용되지 않습니다.

자세한 내용을 확인하십시오

- "알려진 문제"

# 개념

## Astra 데이터 저장소 미리 보기 소개

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

Astra Data Store 미리 보기를 사용하여 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 제품군

Astra 제품군은 Kubernetes 애플리케이션 데이터 라이프사이클 관리를 제공하여 상태 저장 애플리케이션의 운영을 단순화합니다. Kubernetes 워크로드를 손쉽게 보호, 백업, 마이그레이션하고 정상 작동하는 애플리케이션 클론을 즉시 생성할 수 있습니다.

Astra 오퍼링은 다음과 같습니다.

- \* Astra Control \*: 퍼블릭 클라우드와 온프레미스 모두에서 Kubernetes 워크로드를 관리, 보호 및 이동하는 애플리케이션 인식 데이터 관리 툴을 사용합니다
  - \* Astra Control Service \*: NetApp에서 관리하는 서비스를 퍼블릭 클라우드의 Kubernetes 워크로드 데이터 관리에 사용합니다.
  - \* Astra Control Center \*: 사내 Kubernetes 워크로드의 데이터 관리에 자가 관리 소프트웨어를 사용합니다.
- \* Astra Data Store 미리보기 \*: 기업 데이터 관리를 위한 컨테이너 및 VM 워크로드를 위한 Kubernetes 네이티브 공유 파일 서비스를 사용합니다.
- \* Astra Trident \*: NetApp 스토리지 공급자와 함께 CSI(Container Storage Interface)를 준수하는 스토리지 프로비저닝 및 관리 기능을 Kubernetes 워크로드에 사용합니다.

을 참조하십시오 ["Astra 제품군 소개"](#).



## Astra Data Store 미리보기 기능

Astra Data Store 미리 보기에서는 다음과 같은 기능을 사용하여 Kubernetes 네이티브 엔드 스토리지 및 데이터 관리를 클라우드 네이티브 응용 프로그램에 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 데이터 저장소 미리 보기를 시작합니다

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 구축 모델을 미리 봅니다

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

다음 옵션 중 하나를 사용하여 베어 메탈 또는 가상 서버에 Astra Data Store 미리 보기를 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 Store Preview는 Astra 제품군의 일부입니다. Astra 제품군 전체를 보려면 ["Astra 제품군 소개"](#)를 참조하십시오.

## Astra Data Store 미리 보기 에코시스템

Astra Data Store 미리 보기는 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 미리 보기 스토리지 백엔드가 포함된 Astra Trident 스토리지 백엔드를 포함하는 OpenShift 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 Store 미리 보기에서 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 저장소 미리 보기에서 관리되는 리소스에 대한 정보를 표시합니다.

## Astra Data Store 미리 보기 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 미리 보기 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 Store 미리 보기에 대한 정보를 검색할 수도 있습니다.
- **\* kubctl 명령 \***: Astra Data Store 미리 보기와 함께 사용하려면 kudeck 명령을 직접 사용할 수 있습니다.
- **\* Kubernetes 확장 \***: 또한 Astra Data Store 미리보기 Kubernetes API 확장을 사용할 수 있습니다.

사용자 정의 리소스 정의(CRD)는 Astra Data Store 미리보기 운영자가 구축할 때 생성되는 Kubernetes REST API의 확장입니다. 외부 엔터티는 Kubernetes API 서버를 호출하여 CRD와 상호 작용합니다. Astra Data Store 미리 보기는 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 Store 미리보기는 클러스터에서 다양한 유형과 기능을 가진 노드를 지원합니다. 클러스터를 확장하는 경우, Astra Data Store 미리 보기에서는 성능 기능이 있는 노드를 추가할 수 있습니다. 이 경우 클러스터의 성능 중 가장 낮은 노드보다 낮은 노드를 사용할 수 있습니다. 새 노드의 스토리지 용량은 기존 노드와 같아야 합니다. 확장 중에 새 노드를 비롯한 모든 노드는 에서 최소 요구사항을 충족해야 합니다 ["Astra Data Store 미리 보기 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 Store 미리보기 내의 스토리지 효율성

Astra Data Store 미리보기: NetApp ONTAP 및 SolidFire 기술을 기반으로 하는 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 ONTAP 스토리지 시스템은 0으로 제로화된 볼륨의 영역을 감지하여 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. ONTAP는 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 Store 미리 보기의 보안

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

Astra Data Store Preview에서는 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 키 관리를 비롯한 유휴 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 Data Store 미리 보기 요구 사항

먼저 귀사의 환경이 Astra Data Store 미리 보기 요구 사항을 충족하는지 확인하십시오.

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

Astra Data Store preview는 Astra Trident CSI 드라이버를 사용하여 Kubernetes 워크로드에 대한 영구 볼륨의 프로비저닝만 지원합니다. VM 워크로드는 Astra Data Store의 향후 릴리스에서 지원됩니다.



Astra Control Center에서 Astra Data Store 미리 보기 클러스터를 관리하려는 경우 Astra Data Store 미리 보기 클러스터가 에 맞는지 확인하십시오 ["Astra Control Center에서 관리할 클러스터에 대한 요구 사항입니다"](#) 여기에 설명된 요구 사항 외에

### Kubernetes 작업자 노드 리소스 요구사항

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

리소스	최소	최대
데이터 드라이브 수입니다	<ul style="list-style-type: none"><li>3(별도의 캐시 디바이스가 있는 경우)</li><li>4(캐시 디바이스가 없는 경우)</li></ul>	14
데이터 드라이브 크기입니다	100GiB	4TiB
선택적 캐시 디바이스의 수입니다	1(8GiB 이상 크기)	해당 없음
vCPU 수입니다	10	10
RAM	35GiB	35GiB



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 미리 보기 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 요구 사항

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

## 하드웨어 플랫폼

- HPE DL360
- HPE DL380
- Dell R640
- Dell R740

Astra Data Store 미리 보기는 다음 드라이브 유형으로 검증되었습니다.

- \* 베어 메탈 배포 \*: 하이퍼바이저 없이 Kubernetes 클러스터에 직접 설치된 Astra Data Store 미리보기
  - NVMe TLC SSD
- \* VM 기반 구축 \*: ESXi 클러스터에서 호스팅되는 Linux VM의 Kubernetes 클러스터에 설치된 Astra Data Store 미리보기
  - SAS 또는 NVMe TLC SSD 기반 데이터 저장소
  - 드라이브가 가상 디스크 또는 패스스루 드라이브로 표시됩니다



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 Store 미리 보기는 ESXi 7.0을 사용하는 VMware 기반 VM 배포를 통해 검증되었습니다. KVM 기반 배포는 Astra Data Store 미리 보기에서 지원되지 않습니다.
- Astra Data Store 미리 보기는 다음 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 Store Preview는 다음 Kubernetes 배포를 통해 검증되었습니다.
  - Red Hat OpenShift 4.7
  - Google Anthos 1.7
  - Kubernetes 1.21
  - Kubernetes 1.20



Astra Data Store 미리 보기를 사용하려면 스토리지 프로비저닝 및 오케스트레이션을 위해 Astra Trident 버전 21.10.1이 필요합니다. 를 참조하십시오 "[Astra Trident 설치 지침](#)".

## 네트워킹 요구 사항

Astra Data Store 미리 보기를 사용하려면 클러스터당 IP 주소가 하나씩 필요합니다. MIP와 동일한 서브넷에서 사용되지 않거나 구성되지 않은 IP 주소여야 합니다. Astra Data Store 미리 보기 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"><li>• 노드당 2개:<ul style="list-style-type: none"><li>◦ MIP/CIP: 노드당 관리 인터페이스에서 사전 구성된 IP 주소 1개</li><li>◦ 데이터 IP: MIP와 동일한 서브넷의 노드당 사용되지 않거나 구성되지 않은 IP 주소 1개</li></ul></li></ul>
노드당 2개의 네트워크 인터페이스	<ul style="list-style-type: none"><li>• 노드당 3개:<ul style="list-style-type: none"><li>◦ MIP: 노드당 관리 인터페이스에 사전 구성된 IP 주소 1개</li><li>◦ CIP: MIP와 다른 서브넷의 노드별 데이터 인터페이스에 사전 구성된 IP 주소 1개</li><li>◦ 데이터 IP: CIP와 동일한 서브넷에 있는 노드당 사용되지 않거나 구성되지 않은 IP 주소 1개</li></ul></li></ul>



클러스터 사용자 지정 리소스(CR) 파일, "astradscluster.yaml"에서 이 두 가지 구성에 대해 데이터 네트워크 게이트웨이 필드를 생략해야 합니다. 각 노드의 기존 라우팅 구성은 모든 주소를 수용한다.



이러한 구성에는 VLAN 태그가 사용되지 않습니다.

## 아스트라 트리덴트

Astra Data Store 미리 보기를 사용하려면 애플리케이션 Kubernetes 클러스터가 Astra Trident 21.10.1을 실행해야 합니다. Astra Data Store 미리 보기는 로 구성할 수 있습니다 "[스토리지 백엔드](#)" Astra Trident를 사용하여 영구 볼륨 프로비저닝

## CNI 구성

Astra Data Store 미리 보기는 다음 CNIs를 사용하여 검증되었습니다.

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

이러한 CNIs를 사용하려면 호스트 방화벽(firwalld)을 비활성화해야 합니다.

## 지속적인 볼륨 공유 요구 사항

각 Astra Data Store Preview 클러스터는 영구 볼륨을 사용하여 해당 클러스터에 설치된 애플리케이션의 스토리지 요구 사항을 해결할 수 있도록 지원합니다. Astra Data Store 미리 보기의 영구 볼륨에 대한 다음 요구 사항을 고려하십시오.

### 요구 사항

- NFSv4.1 클라이언트/서버는 Kubernetes 클러스터에 설치해야 합니다.
- nfs-utils 패키지를 작업자 노드에 설치해야 합니다.
- Kubernetes 앱은 NFSv4.1을 통해 공유되는 영구 볼륨을 사용하여 파일에 액세스합니다. 이를 위해서는 AUTH\_SYS 인증 방법이 필요합니다.

## 라이센싱

Astra Data Store 미리 보기를 사용하려면 Astra Data Store 미리 보기 라이선스가 있어야 모든 기능을 사용할 수 있습니다. ["여기에서 등록하십시오"](#) Astra Data Store Preview 라이선스를 얻으려면 라이선스 다운로드 지침은 가입 후 발송됩니다.

## AutoSupport 구성

Astra 데이터 저장소 미리 보기를 사용하려면 AutoSupport를 활성화하고 AutoSupport 백엔드에 연결해야 합니다. 이는 직접 인터넷 액세스 또는 프록시 구성을 통해 가능합니다.

를 클릭합니다 ["필수 원격 측정 AutoSupport 번들을 보내는 데 사용되는 주기적 설정입니다"](#) 변경해서는 안 됩니다. 주기적인 AutoSupport 번들 전송을 해제하면 클러스터가 잠기며, 주기적인 설정을 다시 활성화할 때까지 새 볼륨을 생성할 수 없습니다.

## 다음 단계

를 봅니다 ["빠른 시작"](#) 개요.

## Astra 데이터 저장소 미리 보기를 빠르게 시작합니다

이 페이지에서는 Astra Data Store 미리 보기를 시작하는 데 필요한 단계를 개괄적으로 설명합니다. 각 단계의 링크를 클릭하면 자세한 내용을 제공하는 페이지로 이동합니다.

사용해 보세요! Astra Data Store 미리 보기를 사용하려면 90일 미리 보기 라이선스를 사용할 수 있습니다.

["여기에서 등록하십시오"](#) Astra Data Store Preview 라이선스를 얻으려면


Kubernetes 클러스터 요구 사항을 검토합니다

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



- 각 SSD에는 고유한 일련 번호가 있어야 합니다.

에 대해 자세히 알아보십시오 "[Astra Data Store 미리 보기 요구 사항](#)".

 Astra Data Store 미리 보기를 다운로드하고 설치합니다

- 에서 Astra Data Store 미리 보기를 다운로드합니다 "[NetApp Support 사이트](#)".
- 로컬 환경에 Astra Data Store 미리 보기를 설치합니다.
- Astra Data Store Preview 라이선스를 적용합니다.
- Astra Data Store Preview 클러스터를 설치합니다.
- Astra Data Store 미리보기 모니터링을 구성합니다.
- Red Hat OpenShift를 사용하는 경우 Red Hat OpenShift Container Platform(OCP)에 Astra Data Store 미리보기를 설치합니다.

에 대해 자세히 알아보십시오 ["Astra Data Store 미리 보기를 설치하는 중입니다."](#).

`span class="image">&lt;img src="<a href="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png" class="bare">https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png"</a> Alt="3">&lt;/span>` 일부 초기 설정 작업을 완료합니다

- Astra Trident를 설치합니다.
- Kubernetes 스냅샷 CRD(사용자 지정 리소스 정의) 및 컨트롤러를 설치합니다.
- 스토리지 백엔드로 Astra Data Store 미리 보기를 설정합니다.
- 기본 Astra Data Store 미리 보기 스토리지 클래스를 생성합니다.

에 대해 자세히 알아보십시오 ["초기 설정 프로세스"](#).

Astra Data Store 미리 보기 설정을 마치면 다음 작업을 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 Store 미리 보기와 함께 kubelet 명령을 사용하는 방법"](#).
- 모니터링 엔드포인트를 구성합니다. 에 대해 자세히 알아보십시오 ["모니터링 엔드포인트 구성"](#).

"Astra Data Store 미리 보기를 설치합니다".

# Astra Data Store 설치 개요

다음 Astra Data Store 설치 절차 중 하나를 선택하여 완료합니다.

- "표준 프로세스를 사용하여 Astra Data Store를 설치합니다".
- "Red Hat OpenShift를 사용하는 경우 OpenShift를 사용하여 Astra Data Store를 설치합니다".

## Astra Data Store 미리 보기를 설치합니다

Astra Data Store 미리 보기를 설치하려면 에서 설치 번들을 다운로드하십시오 ["NetApp Support 사이트"](#) 및 이 절차에 설명된 설치 단계를 완료합니다.

또는 을(를) 사용할 수 있습니다 ["Red Hat OpenShift Container Platform\(OCP\)에 Astra Data Store Preview 설치"](#).



무엇을 '필요로 할거야

- ["설치를 시작하기 전에 Astra Data Store 미리 보기 구축을 위한 환경을 준비합니다"](#).
- 에 액세스합니다 ["NetApp Support 사이트"](#). 전체 액세스 권한이 없는 NetApp Support 사이트 계정이 없는 경우 미리 보기를 위해
- A ["NetApp 라이선스 파일\(NLF\)"](#) Astra Data Store 미리 보기입니다. 라이선스 다운로드 지침이 사용자에게 전송됩니다 ["가입하세요"](#).
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubeconfig.
- 에 대한 이해 ["역할 및 권한"](#) Astra Data Store 미리 보기에서 사용합니다.
- 인터넷 연결. Astra Data Store 미리 보기는 대기 환경을 지원하지 않습니다. 직접 또는 프록시를 통해 [support.netapp.com](#) 에 접속하려면 인터넷 연결이 필요합니다.

Astra Data Store 미리 보기 설치 프로세스는 다음과 같은 고급 단계를 안내합니다.

- [\[Download the Astra Data Store preview bundle and extract the images\]](#)
- [\[Copy the binary and push images to your local registry\]](#)
- [\[Install the Astra Data Store preview operator\]](#)
- [\[Deploy the Astra Data Store preview version YAML\]](#)
- [\[Apply the Astra Data Store preview license\]](#)

- [\[Install the Astra Data Store preview cluster\]](#)
- [\[Understand deployment-related events\]](#)
- [\[Configure Astra Data Store preview monitoring\]](#)

**Astra Data Store** 미리보기 번들을 다운로드하고 이미지를 추출합니다

1. 에 로그인합니다 "[NetApp Support 사이트](#)" 그리고 Astra Data Store 미리보기 번들('2021.12\_astradatastore.tar')을 다운로드하십시오.
2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. 이미지 추출:

```
tar -xvf 2021.12_astradatastore.tar
```

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

1. k8s kubect 바이너리가 설치된 표준 경로(예: "/usr/bin/")로 이미지를 추출하기 위해 사용한 디렉토리에서 kubectl-astrs 바이너리를 복사합니다. kubectl-astrs는 Astra Data Store Preview 클러스터를 설치 및 관리하는 사용자 지정 kubect 확장입니다.

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

2. Astra Data Store 미리보기 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 미리 보기 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 $(ls images/*.tar) ; do
    astraImage=$(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}'~' ./manifests/*.yaml
```

## Astra Data Store preview operator를 설치한다

### 1. Astra Data Store 미리 보기 목록을 나열합니다.

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

응답:

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

### 2. kubeck 적용 시 운용자 배치:

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

응답:

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
```

customresourcedefinition.apiextensions.k8s.io/astradsllicenses.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.netapp.io created  
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.astrads.netapp.io created  
role.rbac.authorization.k8s.io/astrads-leader-election-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-astradslicense-editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-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-autosupport-editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created  
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created  
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-editor-role created  
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-viewer-role created  
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-

```

role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-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 Preview 버전 YAML을 배포하십시오

### 1. kubectl을 사용하여 구축 적용:

```
kubectl apply -f ./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-astradsversion-2jqnk	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-dbk7v	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-rn9tt	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-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 Preview 라이선스를 적용합니다

1. 미리 보기에 등록할 때 얻은 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	
p100000006	astrads-example-cluster	true	Astra Data Store Preview
true	2022-01-23	2021-11-04T14:38:54Z	

### Astra Data Store Preview 클러스터를 설치합니다

1. YAML 파일을 엽니다.

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

## 2. YAML 파일에서 다음 값을 편집합니다.



YAML 파일의 간단한 예는 다음과 같습니다.

- a. (필수) \* 메타데이터 \*: metadata에서 이름 문자열을 클러스터 이름으로 변경합니다. 이 이름은 사용 시 사용한 클러스터 이름과 같아야 합니다 [라이선스를 적용합니다](#).
- b. (필수) \* Spec \*: 'sepec'에서 다음 필수 값을 변경합니다.
  - 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 미리 보기 노드만큼 데이터 네트워크 IP 주소가 적어도 몇 개 있어야 합니다. Astra Data Store 미리 보기의 경우 4개 이상의 주소가 있어야 하며, 나중에 5개 노드로 클러스터를 확장할 계획이면 5개 이상의 주소가 필요합니다.
  - adsDataNetworks에서 데이터 네트워크에서 사용하는 넷마스크를 지정한다.
  - adsNetworkInterfaces에서 mGMT와 data 값을 관리, 클러스터 및 스토리지에 사용할 네트워크 인터페이스 이름으로 바꿉니다. 이름을 지정하지 않으면 노드의 기본 인터페이스가 관리, 클러스터 및 스토리지 네트워킹에 사용됩니다.



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

- c. (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.



아래의 YAML 샘플에서 대부분의 의견이 제거되었습니다.

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 34
  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 a different interface than
management, specify all three interfaces to use here.
# The Astra Data Store management interface should be same as the
Kubernetes node's management interface.
# NOTE: The cluster and storage networks need to be on the same
interface.

adsNetworkInterfaces:
  managementInterface: "mgmt"
  clusterInterface: "data"
  storageInterface: "data"

# [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.

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

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

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

샘플 반응:

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

5. 클러스터 배포 진행 상태 확인:

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

샘플 반응:

NAME	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster	created	2021.10.0	p100000006	
10.x.x.x	10m			

## 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 메시지입니다	의미
ADS 클러스터를 연결할 4개의 컨트롤 플레인 노드를 성공적으로 선택했습니다	Astra Data Store 미리보기 운영자는 CPU, 메모리, 스토리지 및 네트워킹으로 Astra Data Store 미리보기 클러스터를 생성할 수 있는 충분한 노드를 식별했습니다.
ADS 클러스터 생성 진행 중	Astra Data Store 미리보기 클러스터 컨트롤러가 클러스터 생성 작업을 시작했습니다.
ADS 클러스터가 생성되었습니다	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 Data Store 미리보기 모니터링을 구성합니다

Astra Control Center 모니터링 또는 다른 원격 측정 서비스의 모니터링을 위해 Astra Data Store 미리보기를 구성할 수 있습니다.

### Astra Control Center 미리 보기에 대한 모니터링을 구성합니다

Astra Control Center에서 Astra Data Store 미리 보기를 백엔드로 관리하는 경우에만 다음 단계를 수행하십시오.

1. Astra Control Center에서 모니터링하는 Astra Data Store 미리 보기를 구성합니다.

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

모니터링 운전자를 설치합니다

(선택 사항) Astra Data Store Preview를 Astra Control Center로 가져오지 않는 경우에만 모니터링 운영자를 권장합니다. Astra 데이터 저장소 미리 보기 인스턴스가 독립 실행형 배포이거나, Cloud Insights를 사용하여 원격 측정을 모니터링하거나, Elastic과 같은 타사 엔드포인트로 로그를 스트리밍하는 경우 모니터링 연산자를 설치할 수 있습니다.

1. 다음 설치 명령을 실행합니다.

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

2. 모니터링을 위해 Astra Data Store 미리 보기를 구성합니다.

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

다음 단계

를 수행하여 배포를 완료합니다 **"설정 작업"**.

## Red Hat OpenShift Container Platform에 Astra Data Store 미리보기를 설치합니다

Red Hat OpenShift Container Platform(OCP)에 Astra Data Store 미리보기를 설치하려면 에서 설치 번들을 다운로드하십시오 **"NetApp Support 사이트"** 및 이 절차에 설명된 설치 단계를 완료합니다.

무엇을 '필요로 할거야

- 설치를 시작하기 전에 **"Astra Data Store 구축을 위한 환경을 준비합니다"**.
- 에 액세스합니다 **"NetApp Support 사이트"**. 전체 액세스 권한이 없는 NetApp Support 사이트 계정이 없는 경우 미리 보기를 위해
- A **"NetApp 라이선스 파일"** (NLF) - Astra Data Store 미리 보기. 라이선스 다운로드 지침은 가입 후 발송됩니다.
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubeconfig.
- 에 대한 이해 **"역할 및 권한"** Astra Data Store 미리 보기에서 사용합니다.
- 인터넷 연결. Astra Data Store Preview는 대기 환경을 지원하지 않습니다. 직접 또는 프록시를 통해 support.netapp.com 에 접속하려면 인터넷 연결이 필요합니다.

이 작업에 대해

Astra Data Store 미리 보기 설치 프로세스는 다음과 같은 고급 단계를 안내합니다.

- [\[Download the Astra Data Store preview bundle and extract the images\]](#)
- [\[Copy the binary and push images to your local registry\]](#)
- [\[Create a namespace to deploy Astra Data Store preview\]](#)
- [\[Create a custom SCC\]](#)
- [\[Create the roles and role bindings\]](#)

- [\[Install the Astra Data Store preview operator\]](#)
- [\[Deploy the Astra Data Store preview version YAML\]](#)
- [\[Apply the Astra Data Store preview license\]](#)
- [\[Install the Astra Data Store preview cluster\]](#)
- [\[Understand deployment-related events\]](#)
- [\[Configure Astra Data Store preview monitoring\]](#)
- [\[Install the monitoring operator\]](#)
- [\[Set up iptables\]](#)

**Astra Data Store** 미리보기 번들을 다운로드하고 이미지를 추출합니다

1. 에 로그인합니다 "[NetApp Support 사이트](#)" 그리고 Astra Data Store 미리보기 번들('2021.12\_astradatastore.tar')을 다운로드하십시오.
2. (선택 사항) 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. 이미지 추출:

```
tar -xvf 2021.12_astradatastore.tar
```

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

1. k8s kubect 바이너리가 설치된 표준 경로(예: "/usr/bin/")로 이미지를 추출하기 위해 사용한 디렉토리에서 kubectl-astrs 바이너리를 복사합니다. kubectl-astrs는 Astra Data Store Preview 클러스터를 설치 및 관리하는 사용자 지정 kubctl 확장입니다.

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

2. Astra Data Store 미리보기 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 미리 보기 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 $(ls images/*.tar) ; do
    astraImage=$(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}'~' ./manifests/*.yaml
```

**Astra Data Store** 미리 보기를 배포할 네임스페이스를 생성합니다

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

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
```

사용자 지정 **SCC**를 생성합니다

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 미리보기 클러스터 포드에 필요한 권한을 제공하지 않습니다. 이 절차를 사용하여 Astra Data Store 미리 보기에 필요한 권한(샘플에 나열되어 있음)을 제공합니다.

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

단계

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                    ["*"]
#
```

역할 및 역할 바인딩을 만듭니다

Astra Data Store 미리 보기의 기본 서비스 계정에서 사용할 필수 역할 및 역할 바인딩을 만듭니다.

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

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

```

## 작업자 노드를 준비합니다

Astra Data Store 클러스터 미리 보기 구축을 위해 작업자 노드를 준비합니다. Astra Data Store preview 클러스터에 사용되는 모든 작업자 노드에서 이 절차를 수행합니다.

OpenShift는 kubelet 구성 파일('/var/lib/kubelet/config.json')에 json 형식을 사용합니다. Astra Data Store Preview 클러스터는 kubelet config 파일의 YAML 형식을 찾습니다.

## 단계

1. 클러스터 설치를 시작하기 전에 각 작업자 노드에 '/var/lib/kubelet/config.YAML' 파일을 생성합니다.

```
sudo cp /var/lib/kubelet/config.json /var/lib/kubelet/config.yaml`
```

2. 클러스터 YAML이 적용되기 전에 모든 Kubernetes 노드에서 이 절차를 완료하십시오.



이렇게 하지 않으면 Astra Data Store Preview 클러스터 설치가 실패합니다.

### Astra Data Store preview operator를 설치한다

1. Astra Data Store 미리 보기 목록을 나열합니다.

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

응답:

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

2. kubectl apply 명령을 사용하여 운영자를 배치한다.

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

응답:

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
```

```
.netapp.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-leader-election-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-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-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-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
```

```
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-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 Preview 버전 YAML을 배포하십시오

### 1. kubbeck apply 명령을 사용하여 구축:

```
kubectl apply -f ./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-astradsversion-2jqnk	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-dbk7v	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-rn9tt	1/1	Running	0
2m7s			
astrads-ds-nodeinfo-astradsversion-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 Preview 라이선스를 적용합니다

1. 미리 보기에 등록할 때 얻은 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	
p100000006	astrads-example-cluster	true	Astra Data Store Preview
true	2022-01-23	2021-11-04T14:38:54Z	

### Astra Data Store Preview 클러스터를 설치합니다

1. YAML 파일을 엽니다.

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

## 2. YAML 파일에서 다음 값을 편집합니다.



YAML 파일의 간단한 예는 다음과 같습니다.

- a. (필수) \* 메타데이터 \*: metadata에서 이름 문자열을 클러스터 이름으로 변경합니다. 이 이름은 사용 시 사용한 클러스터 이름과 같아야 합니다 [라이선스를 적용합니다](#).
- b. (필수) \* Spec \*: 'sepec'에서 다음 필수 값을 변경합니다.
  - 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 미리 보기 노드만큼 데이터 네트워크 IP 주소가 적어도 몇 개 있어야 합니다. Astra Data Store 미리 보기의 경우 4개 이상의 주소가 있어야 하며, 나중에 5개 노드로 클러스터를 확장할 계획이면 5개 이상의 주소가 필요합니다.
  - adsDataNetworks에서 데이터 네트워크에서 사용하는 넷마스크를 지정한다.
  - adsNetworkInterfaces에서 mGMT와 data 값을 관리, 클러스터 및 스토리지에 사용할 네트워크 인터페이스 이름으로 바꿉니다. 이름을 지정하지 않으면 노드의 기본 인터페이스가 관리, 클러스터 및 스토리지 네트워킹에 사용됩니다.



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

- c. (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.



아래의 YAML 샘플에서 대부분의 의견이 제거되었습니다.

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 34
  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 a different interface than
management, specify all three interfaces to use here.
# The Astra Data Store management interface should be same as the
Kubernetes node's management interface.
# NOTE: The cluster and storage networks need to be on the same
interface.

adsNetworkInterfaces:
  managementInterface: "mgmt"
  clusterInterface: "data"
  storageInterface: "data"

# [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.

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

4. SELinux가 설정된 경우 Astra Data Store preview 클러스터의 노드에 있는 다음 디렉토리에 대한 'strinux' 컨텍스트의 레이블을 다시 지정합니다.

```
sudo chcon -R -t container_file_t  
/var/opt/netapp/firetap/rootfs/var/asup/notification/firetap/
```

```
sudo chcon -R -t container_file_t /var/netapp/firetap/firegen/persist/
```



이 단계는 셀린스가 이 디렉토리를 쓸 수 없도록 하여 지원 포드가 CrashLoopoff 상태로 들어가는 원인이 되기 때문에 필요합니다. 이 단계는 Astra Data Store 미리 보기 클러스터의 모든 노드에서 수행해야 합니다.

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

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

샘플 반응:

```
NAME READY STATUS RESTARTS AGE  
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m  
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d  
astrads-ds-astrads-cluster-lab0dbc-j9jzc 1/1 Running 0 5d2h  
astrads-ds-astrads-cluster-lab0dbc-k9wp8 1/1 Running 0 5d1h  
astrads-ds-astrads-cluster-lab0dbc-pwk42 1/1 Running 0 5d2h  
astrads-ds-astrads-cluster-lab0dbc-qhvc6 1/1 Running 0 8h  
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d  
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d  
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d  
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d  
astrads-ds-support-828vw 2/2 Running 2 5d2h  
astrads-ds-support-cfzts 2/2 Running 0 8h  
astrads-ds-support-nzkkkr 2/2 Running 15 7h49m  
astrads-ds-support-xxbnp 2/2 Running 1 5d2h  
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h  
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

6. 클러스터 배포 진행 상태 확인:



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

샘플 반응:

NAME	STATUS	VERSION	SERIAL NUMBER	MVIP
astrads-example-cluster	created	2021.10.0	p1000000006	
10.x.x.x	10m			

배포 관련 이벤트를 이해합니다

클러스터 배치 중에는 작동 상태가 공란에서 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 메시지입니다	의미
ADS 클러스터를 연결할 4개의 컨트롤 플레인 노드를 성공적으로 선택했습니다	Astra Data Store 미리보기 운영자는 CPU, 메모리, 스토리지 및 네트워킹으로 Astra Data Store 미리보기 클러스터를 생성할 수 있는 충분한 노드를 식별했습니다.
ADS 클러스터 생성 진행 중	Astra Data Store 미리보기 클러스터 컨트롤러가 클러스터 생성 작업을 시작했습니다.
ADS 클러스터가 생성되었습니다	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 Data Store 미리보기 모니터링을 구성합니다

Astra Control Center 모니터링 또는 다른 원격 측정 서비스의 모니터링을 위해 Astra Data Store 미리보기를 구성할 수 있습니다.

### Astra Control Center 미리 보기에 대한 모니터링을 구성합니다

Astra Control Center에서 Astra Data Store 미리 보기를 백엔드로 관리하는 경우에만 다음 단계를 수행하십시오.

1. Astra Control Center에서 모니터링하는 Astra Data Store 미리 보기를 구성합니다.

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

모니터링 운전자를 설치합니다

(선택 사항) Astra Data Store Preview를 Astra Control Center로 가져오지 않는 경우에만 모니터링 운영자를 권장합니다. Astra 데이터 저장소 미리 보기 인스턴스가 독립 실행형 배포이거나, Cloud Insights를 사용하여 원격 측정을 모니터링하거나, Elastic과 같은 타사 엔드포인트로 로그를 스트리밍하는 경우 모니터링 연산자를 설치할 수 있습니다.

1. 다음 설치 명령을 실행합니다.

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

2. 모니터링을 위해 Astra Data Store 미리 보기를 구성합니다.

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

## iptables를 설정합니다

OpenShift SDN(OpenShift용 기본 CNI 플러그인)은 호스트 루프백 인터페이스(127.0.0.1/localhost)에서 HostPorts로 트래픽을 마스커하지 않습니다.

Astra Data Store 미리보기 클러스터에는 localhost에서 클러스터 서비스 포트(9920)로 트래픽을 전달하기 위한 NAT 규칙이 추가되어 있어야 합니다.

단계

1. NAT 테이블의 출력 체인에 있는 현재 KUBE-HOSTS 규칙의 행 번호를 확인합니다.

아래 예에서는 KUBE-HOSTS가 4위치에 있습니다.

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals */
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE: this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type LOCAL /* handle service NodePorts; NOTE: this must be the last rule in the chain */
4 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */ ADDRTYPE match dst-type LOCAL
```

## 2. KUBE-HOSTS 항목 바로 위에 NAT 테이블에 새 규칙을 추가합니다.

```
$ sudo iptables -t nat -I OUTPUT 4 -s 127.0.0.1 -j KUBE-MARK-MASQ -p tcp --dport 9920
```

## 3. 새로 추가된 규칙이 KUBE-HOSTS 규칙 바로 위에 NAT 테이블에 추가되었는지 확인합니다.

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals */
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE: this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type LOCAL /* handle service NodePorts; NOTE: this must be the last rule in the chain */
4 KUBE-MARK-MASQ tcp -- localhost anywhere tcp dpt:9920
5 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */ ADDRTYPE match dst-type LOCAL
```



Astra Data Store 미리보기 클러스터의 모든 노드에 대해 단계를 수행합니다.



Astra Data Store preview 클러스터 노드가 다시 시작되면 이 단계를 반복합니다.

다음 단계

를 수행하여 배포를 완료합니다 **"설정 작업"**.

# Astra Data Store 미리보기 구성 요소를 설정합니다

Astra Data Store 미리 보기를 설치하고 몇 가지 환경 요구 사항을 해결한 후에는 Astra Trident를 설치하고, Kubernetes 스냅샷 기능을 구성하고, 스토리지 백엔드를 설정하고, 기본 스토리지 클래스를 생성합니다.

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

## Astra Trident를 설치합니다

Astra Data Store 미리 보기의 경우 Astra Trident 21.10.1을 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 미리 보기 구축을 위한 환경을 준비합니다"](#).
- 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 created
```

## 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 created
```

## 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.storage.k8s.io created
```

## 4. 설정 적용 - 스냅샷 - 컨트롤러.YAML:

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

응답:

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

## 5. RBAC-스냅샷-컨트롤러 적용.YAML:

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

응답:

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

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:

- a. ""클러스터"" 값을 Astra Data Store 클러스터의 클러스터 이름으로 변경합니다.
- b. "namespace" 값을 볼륨 생성에 사용할 네임스페이스로 변경합니다.
- c. 이 백엔드에 대한 내보내기 정책 CR을 대신 설정하지 않는 한 ""autoExportPolicy"" 값을 "true"로 변경합니다.
- d. 액세스를 허용할 IP 주소로 ""autoExportCIDR"" 목록을 채웁니다. 모두 허용하려면 0.0.0.0/0을 사용하십시오.
- e. "kubecononfig" 값을 보려면 다음을 수행합니다.

- i. 공백 없이 .kubbe/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": "bronze"
  },
  "storage": [
    {
      "labels": {
        "performance": "extreme"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    },
    {
      "labels": {
        "performance": "premium"
      },
      "defaults": {
        "qosPolicy": "bronze",
      }
    },
    {
      "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
```

응답:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
mountOptions:
  - vers=4

```

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 -A
```

응답:

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION	AGE		
trident-csi	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: bronze
          size: ""
          snapshotDir: ""
          snapshotPolicy: ""
        labels:
          performance: extreme
          region: ""
          supportedTopologies: null
          zone: ""
      - defaults:
          exportPolicy: ""
          qosPolicy: bronze
          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:
  - trident-csi
  supportedTopologies: null
volumes: []

```

# Astra Data Store 미리보기 제한

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 Store 미리보기 릴리즈에는 다음과 같은 리소스 제한이 있습니다.

리소스	최소	최대
Astra Data Store 미리보기 클러스터의 노드 수입니다	4	5
노드당 영구 볼륨 수입니다	해당 없음	10
노드당 영구 볼륨의 프로비저닝된 총 용량입니다	해당 없음	1TiB
볼륨 크기	20MiB	1TiB
볼륨당 스냅샷	0	256
볼륨당 클론	0	9



Astra Data Store 미리 보기는 VM 워크로드를 지원하지 않습니다. VMware VVOL 워크로드 지원은 향후 릴리즈에서 제공될 예정입니다.



Astra Data Store 미리 보기는 성능 임계치를 조절하며 성능 특성화에 사용해서는 안 됩니다.

## 다음 단계

구성이 에 맞는지 확인합니다 ["요구 사항"](#).

## Astra Data Store 미리 보기를 위한 질문과 대답

Astra Data Store 미리 보기의 설치, 구성, 업그레이드 및 문제 해결에 대해 자주 묻는 질문에 대한 답변을 찾아보십시오.

### 일반적인 질문

- 생산에 Astra Data Store 미리보기를 사용할 수 있습니까? \* 아니요 Astra Data Store는 엔터프라이즈급 복구 성능을 제공하도록 설계 및 개발되었지만, 미리 보기 릴리즈인 Astra Data Store Preview는 운영 워크로드를 대상으로 하지 않습니다.
- 가상 머신 워크로드에 Astra Data Store 미리보기를 사용할 수 있습니까? \* Astra Data Store 미리보기 릴리즈는 베어 메탈 또는 가상 머신에서 실행 중인 애플리케이션에만 한정됩니다. 향후 릴리즈에서는 Kubernetes 및 ESXi 가상 머신 모두에서 애플리케이션을 직접 지원합니다. 을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#).
- Astra Data Store 미리 보기에는 다른 NetApp 제품의 기능에 대한 의존성이 있습니까? \*

예. Astra Data Store 미리 보기를 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"](#).

Astra Data Store Preview 클러스터를 스토리지 백엔드로 사용하는 애플리케이션은 를 사용할 수 있습니다 ["Astra"](#)

제어 센터" 버전 21.12를 사용하여 Kubernetes 워크로드의 데이터 보호, 재해 복구, 마이그레이션을 비롯한 애플리케이션 인식 데이터 관리 기능을 활용할 수 있습니다.

- Astra Data Store 미리보기 클러스터를 관리하려면 어떻게 해야 하나요? \* kubctl 명령을 사용하거나 Kubernetes API 확장을 사용하여 Astra Data Store 미리보기 자산을 관리할 수 있습니다.

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

- Astra 데이터 저장소 미리 보기 클러스터 메트릭을 어떻게 모니터링할 수 있습니까? \* Cloud Insights를 사용하여 Astra 데이터 저장소 미리 보기 메트릭을 모니터링할 수 있습니다. 을 참조하십시오 ["Cloud Insights를 통해 메트릭을 모니터링합니다"](#).

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

- Kubernetes 클러스터의 ONTAP 또는 다른 스토리지 공급자와 함께 Astra Data Store 미리보기를 사용할 수 있습니까? \* 예. Astra Data Store 미리 보기는 애플리케이션 클러스터의 다른 스토리지 공급자와 함께 사용할 수 있습니다.
- Astra 데이터 저장소 미리 보기에서 Kubernetes 클러스터를 제거하면 Astra Trident가 제거됩니까? \* Astra 데이터 저장소 미리 보기를 제거하면 Astra Trident가 클러스터에서 제거되지 않습니다. Astra Trident를 제거해야 하는 경우 별도로 제거해야 합니다.

## 라이센싱

- Astra Data Store 미리 보기에는 라이선스가 필요합니까? \* 예, Astra Data Store 미리 보기를 사용하려면 NetApp 라이선스 파일(NLF)이 필요합니다.

을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#).

- Astra 데이터 저장소 미리 보기 라이선스는 얼마나 유효합니까? \* Astra 데이터 저장소 미리 보기 라이선스의 기본 기간은 다운로드 날짜로부터 90일입니다.

## Kubernetes 클러스터에서 Astra Data Store 미리 보기의 설치 및 사용

- 베어 메탈 또는 가상 머신에서 실행 중인 Kubernetes 클러스터에 Astra Data Store 미리보기를 설치할 수 있습니까? \* 예. Astra Data Store 미리 보기는 베어 메탈 또는 ESXi VM에서 실행되는 Kubernetes 클러스터에 설치할 수 있습니다. 을 참조하십시오 ["Astra Data Store 미리 보기 요구 사항"](#).
- Astra Data Store Preview용 Kubernetes의 지원되는 버전은 무엇입니까? \*

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

- My Kubernetes 클러스터가 5개 이상의 작업자 노드입니다. Astra 데이터 저장소 미리 보기를 설치할 수 있습니까? \* 예. Kubernetes 클러스터의 4개 작업자 노드에 Astra Data Store 미리보기 클러스터를 구축할 수 있습니다. 구축한 후에는 클러스터를 5개의 작업자 노드로 확장할 수 있습니다.
- Astra 데이터 저장소 미리 보기는 개인 레지스트리의 오프라인 설치를 지원합니까? \* 예. Astra Data Store 미리 보기는 로컬 레지스트리에서 오프라인으로 설치할 수 있습니다. 을 참조하십시오 ["Astra Data Store 미리 보기를 설치합니다"](#). 그러나 Astra Data Store 미리 보기를 사용하려면 NetApp AutoSupport 백엔드(support.netapp.com 직접 또는 프록시를 통해 연결해야 합니다).
- Astra 데이터 저장소 미리 보기를 사용하려면 인터넷 연결이 필요합니까? \* Astra 데이터 저장소 미리 보기를 사용하려면 NetApp AutoSupport 백엔드에 연결해야 필수 원격 측정 AutoSupport 번들을 정기적으로 보낼 수 있습니다. 이러한 연결은 직접 또는 프록시를 통해 가능합니다. 이 연결이 없거나 AutoSupport가 비활성화된 경우

클러스터가 잠기며 주기적인 번들 업로드가 재개될 때까지 새 볼륨 생성이 비활성화됩니다.

- Astra Data Store Preview에서 사용하는 역할과 권한은 무엇입니까? \* Astra Data Store Preview 운영자를 배포하려면 kudo 관리자여야 합니다.

Astra Data Store Preview에는 노드 선택에 사용되는 노드 리소스를 검색하는 데 사용되는 "astrads-ds-nodeinfo-astradsversion"이라는 특별한 데모가 포함되어 있습니다.

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

- Astra Data Store 미리 보기 설치를 위해 업데이트해야 하는 매니페스트 파일은 무엇입니까? \* 에서 다운로드한 Astra Data Store 미리 보기 번들에서 다운로드할 수 있습니다 ["NetApp Support 사이트"](#), 다음과 같은 매니페스트가 제공됩니다.
- Astasscluster.YAML
- Astassoperator.YAML
- Astradsversion.YAML을 참조하십시오
- monitoring\_operator.YAML

배포별 구성으로 "astradscluster.yaml" 매니페스트를 업데이트해야 합니다. 을 참조하십시오 ["Astra Data Store 미리 보기를 설치합니다"](#).

## 문제 해결 및 지원

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

### ["NetApp 컨테이너 여유 채널"](#)

Preview 릴리즈를 사용하려면 시스템이 클라우드에 연결되고 NetApp Active IQ 및 AutoSupport 도구에 통합되어 있어야 합니다.

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](mailto:astra.feedback@netapp.com) 으로 문의하십시오.
- 지원 로그 번들을 생성하려면 어떻게 합니까? \* 를 참조하십시오 ["지원 번들을 생성합니다"](#) Astra Data Store 미리 보기를 위한 지원 로그 번들을 설정 및 다운로드하는 방법에 대한 지침은 을 참조하십시오.
- Astra Data Store 미리 보기에서 내 Kubernetes 노드를 찾을 수 없습니다. 이 문제를 해결하려면 어떻게 합니까? \* 를 참조하십시오 ["Astra Data Store 미리 보기를 설치합니다"](#).
- IPv6 주소를 관리, 데이터 및 클러스터 네트워크에 사용할 수 있습니까? \* 아니요. Astra Data Store 미리 보기는 IPv4 주소만 지원합니다. IPv6 지원은 Astra Data Store Preview의 향후 릴리스에 추가될 예정입니다.
- Astra Data Store 미리 보기에서 볼륨을 프로비저닝하는 동안 어떤 NFS 버전이 사용됩니까? \* 기본적으로 Astra Data Store 미리 보기는 Kubernetes 애플리케이션에 프로비저닝된 모든 볼륨에 대해 NFS v4.1을 지원합니다.
- 대용량 드라이브를 사용하여 Astra Data Store 미리보기를 구성한 경우에도 더 큰 영구 볼륨을 얻을 수 없는 이유는



무엇입니까? \* Astra Data Store 미리 보기는 노드의 모든 볼륨에 대해 프로비저닝된 최대 용량을 1TiB로 제한하고 Astra Data의 모든 노드에서 최대 5TiB로 제한합니다 Store preview cluster(미리보기 클러스터 저장)

을 참조하십시오 "[Astra Data Store 미리 보기 요구 사항](#)".

## **Astra Data Store** 미리 보기를 업그레이드하는 중입니다

- Astra Data Store Preview 릴리스에서 업그레이드할 수 있습니까? \* 아니요 Astra Data Store 미리 보기는 운영 작업 부하가 아니며, Astra Data Store 미리 보기 소프트웨어의 새 릴리즈에는 새로 설치해야 합니다.

# Astra Data Store를 사용하십시오

## kubectl 명령을 사용하여 Astra Data Store 미리 보기 자산을 관리합니다

kubectl 명령을 사용하거나 Kubernetes API 확장을 사용하여 Astra Data Store 미리보기 자산을 관리할 수 있습니다.

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

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

- 노드를 유지보수 모드로 전환합니다
- 드라이브를 교체합니다
- 노드를 추가합니다
- 노드를 교체합니다

무엇을 '필요로 할거야

- 에 설치한 Astra Data Store preview kubectl 플러그인 ["Astra Data Store 미리 보기를 설치합니다"](#)

## Astra Data Store 미리보기에 대한 Kubernetes 사용자 지정 API 리소스를 나열합니다

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

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

이 목록은 나중에 표시된 것처럼 입력을 줄이기 위해 각 Astra Data Store 미리 보기 개체의 단축 이름을 가져오는 데 특히 유용합니다.

1. Astra Data Store 미리 보기를 위한 Kubernetes 사용자 정의 API 리소스 목록을 표시합니다.

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

응답:

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
astradsversions	adsve	astrads.netapp.io	true	
AstraDSVersion				
astradsclusters	adscl	astrads.netapp.io	true	
AstraDSCluster				
astradslicenses	adsli	astrads.netapp.io	true	
AstraDSLICENSE				
astradsnodeinfoes	adsni	astrads.netapp.io	true	
AstraDSNodeInfo				
astradsvolumes	adsvo	astrads.netapp.io	true	
AstraDSVolume				
astradsqospolicies	adsqp	astrads.netapp.io	true	
AstraDSQosPolicy				
astradsexportpolicies	adsep	astrads.netapp.io	true	
AstraDSEExportPolicy				
astradsvolumesnapshots	adsvs	astrads.netapp.io	true	
AstraDSVolumeSnapshot				
astradsvolumefiles	adsvf	astrads.netapp.io	true	
AstraDSVolumeFiles				
astradsautosupports	adsas	astrads.netapp.io	true	
AstraDSAutoSupport				
astradsfaileddrives	adsfd	astrads.netapp.io	true	
AstraDSFailedDrive				
astradsnodemanagements	adsnm	astrads.netapp.io	true	
AstraDSNodeManagement				

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

```
kubectl 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/	created	arda-9.11.1	e000000009	10.224.8.146	46h

```

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     astradslicense.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
astrads-system	test234	21Gi	172.25.138.109	astrads-cluster-9f1c99f
astrads-system	test2345	21Gi	172.25.138.111	astrads-cluster-9f1c99f

## kubeck 확장명에 대한 도움말 옵션을 사용합니다

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

1. Astra Data Store preview kubctl extension의 모든 명령에 대한 도움말을 표시합니다.

```
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:

--as string	Username to impersonate for the operation
--as-group stringArray	Group to impersonate for the operation

operation, this flag can be

groups.

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

<pre>-c, --component string</pre>	Specify the component to collect: [storage , controlplane , vasaprovider, all]
<pre>-d, --duration int</pre>	Duration is the duration in hours from the startTime for collection
<pre>-e, --event string</pre>	of AutoSupport. This should be a positive integer
<pre>(default "manual")</pre>	Specify the callhome event to trigger.
<pre>-f, --forceUpload</pre>	Configure an AutoSupport to upload if
<pre>it is in the compressed state</pre>	and not
	uploading because it was created with
<pre>the 'local' option or if</pre>	automatic uploads of AutoSupports is
<pre>disabled</pre>	at the cluster level.
<pre>-h, --help</pre>	help for collect
<pre>-l, --local</pre>	Only collect and compress the
<pre>autosupport bundle. Do not upload</pre>	to support.
	Use 'download' to copy the collected

```

bundle after it is in
--nodes string          the 'compressed' state
                        Specify nodes to collect for storage
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

Astra Data Store 미리 보기와 함께 사용할 수 있는 테스트 응용 프로그램을 배포하는 단계는 다음과 같습니다.

이 예에서는 Hrom 리포지토리를 사용하여 Bitnami의 MongoDB 차트를 배포합니다.

무엇을 '필요로 할거야

- Astra Data Store 미리보기 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 명령 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

```

## 클러스터를 관리합니다

Astra Data Store 미리 보기와 함께 kubctl 명령을 사용하여 클러스터를 관리할 수 있습니다.

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

무엇을 '필요로 할거야

- kubctl 및 kubbctl-astrads 플러그인이 설치된 시스템. 을 참조하십시오 ["Astra Data Store 미리 보기를 설치합니다"](#).

## 노드를 추가합니다

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

단계

1. 새 노드의 데이터 IP가 ADSCluster CR에 아직 포함되지 않은 경우 다음을 수행합니다.
  - a. astradscluster CR을 편집하고 ADS 데이터 네트워크 주소 필드에 추가 데이터 IP를 추가합니다.

```
~% kubectl edit astradscluster <cluster-name> -n astrads-system
```

응답:

ADS Data Networks:

Addresses: dataIP1, dataIP2, dataIP3, dataIP4, \*newdataIP\*

- a. CR 파일을 저장합니다.
- b. Astra Data Store 미리 보기 클러스터에 노드를 추가합니다.

```
~% kubectl astrads nodes add -cluster <cluster-name>
```

2. 그렇지 않으면 노드를 추가하기만 하면 됩니다.

```
~% kubectl astrads nodes add -cluster <cluster-name>
```

3. 노드가 추가되었는지 확인합니다.

```
~% kubectl astrads nodes list
```

## 노드를 유지보수 모드로 전환합니다

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



노드는 이미 Astra Data Store 미리 보기 클러스터에 포함되어 있어야 합니다.

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

```
~% kubectl astrads maintenance create <cr-name> --node-name=<<node-name>> --variant=Node
```

샘플:

```
~% kubectl astrads maintenance create maint1 --node-name="nhcitjj1525"
--variant=Node
Maintenance mode astrads-system/maint1 created
```

4. 노드를 나열합니다.

```
~% 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
```

## 노드를 교체합니다

Astra Data Store 미리 보기와 함께 kubectl 명령을 사용하여 클러스터에서 장애가 발생한 노드를 교체합니다.

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. 클러스터 설명:

```
~% kubectl astrads clusters list
```

응답:

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

3. 노드 HA가 장애가 발생한 노드에서 FALSE로 표시되는지 확인합니다.

```
~% kubectl describe astradscluster -n astrads-system
```

응답:

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"astrads.netapp.io/v1alpha1","kind":"AstraDSCluster","meta
data":{"annotations":{},"name":"cluster-multinodes-21209","namespa...
API Version:   astrads.netapp.io/v1alpha1
Kind:          AstraDSCluster

State:          Disabled
Variant:        None
Node HA:        false
Node ID:        4
Node Is Reachable: false
Node Management IP: 172.21.192.192
Node Name:      sti-rx2540-532dctl.gdl.englab.netapp.com
Node Role:      Storage
Node UUID:      6f6b88f3-8411-56e5-b1f0-a8e8d0c946db
Node Version:   12.75.0.6167444
Status:         Added
```

4. "AdsNode Count"의 값을 3으로 감소함으로써 실패한 노드를 제거하도록 Astra클러스터 CR을 수정합니다.

```
cat manifests/astradscluster.yaml
```

응답:

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
```



```

name: cluster-multinodes-21209
namespace: astrads-system
spec:
  # ADS Node Configuration per node settings
  adsNodeConfig:
    # Specify CPU limit for ADS components
    # Supported value: 9
    cpu: 9
    # Specify Memory Limit in GiB for ADS Components.
    # Your kubernetes worker nodes need to have at least this much RAM
    free
    # for ADS to function correctly
    # Supported value: 34
    memory: 34
    # [Optional] Specify raw storage consumption limit. The operator
    will only select drives for a node up to this limit
    capacity: 600
    # [Optional] Set a cache device if you do not want auto detection
    e.g. /dev/sdb
    # cacheDevice: ""
    # Set this regex filter to select drives for ADS cluster
    # drivesFilter: ".*"

    # [Optional] Specify node selector labels to select the nodes for
    creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue

    # Specify the number of nodes that should be used for creating ADS
    cluster
    adsNodeCount: 3

    # Specify the IP address of a floating management IP routable from any
    worker node in the cluster
    mvip: "172..."

    # Comma separated list of floating IP addresses routable from any host
    where you intend to mount a NetApp Volume
    # at least one per node must be specified
    # addresses: 10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.5
    # netmask: 255.255.255.0
    adsDataNetworks:
      - addresses: "172..."
        netmask: 255.255.252.0

```

```

# [Optional] Provide a k8s label key that defines which protection
domain a node belongs to
# adsProtectionDomainKey: ""

# [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
monitoringConfig:
  namespace: "netapp-monitoring"
  repo: "docker.repo.eng.netapp.com/global/astra"

autoSupportConfig:
  # AutoUpload defines the flag to enable or disable AutoSupport
  upload in the cluster (true/false)
  autoUpload: true
  # Enabled defines the flag to enable or disable automatic
  AutoSupport collection.
  # When set to false, periodic and event driven AutoSupport
  collection would be disabled.
  # It is still possible to trigger an AutoSupport manually while
  AutoSupport is disabled
  # enabled: true
  # CoredumpUpload defines the flag to enable or disable the upload of
  coredumps for this ADS Cluster
  # coredumpUpload: false
  # HistoryRetentionCount defines the number of local (not uploaded)
  AutoSupport Custom Resources to retain in the cluster before deletion
  historyRetentionCount: 25
  # DestinationURL defines the endpoint to transfer the AutoSupport
  bundle collection
  destinationURL: "https://testbed.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  # Periodic defines the config for periodic/scheduled AutoSupport
  objects
  periodic:
    # Schedule defines the Kubernetes Cronjob schedule
    - schedule: "0 0 * * *"
    # PeriodicConfig defines the fields needed to create the
    Periodic AutoSupports
    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
```

5. 클러스터에서 노드가 제거되었는지 확인합니다.

```
~% kubectl get nodes --show-labels
```

응답:

NAME	STATUS	ROLES	AGE	VERSION
LABELS				
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
sti-rx2540-532d	Ready	<none>	24h	v1.20.0
sti-rx2540-533d	Ready	<none>	24h	

```
~% 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
sti-rx2540-536d	Added	cluster-multinodes-21209

```
~% kubectl get nodes --show-labels
```

응답:

NAME	STATUS	ROLES	AGE	VERSION
LABELS				
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
sti-rx2540-532d	Ready	<none>	24h	v1.20.0
beta.kubernetes.io/arch=amd64, astrads.netapp.io/node-removal				

```
~% kubectl describe astradscluster -n astrads-system
```

응답:

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Kind:          AstraDSCluster
Metadata:
...
```

6. 클러스터 CR을 수정하여 교체할 노드를 클러스터에 추가합니다. 노드 수는 4까지 증가합니다. 추가를 위해 새 노드가 선택되었는지 확인합니다.

```
rvi manifests/astradscluster.yaml
cat manifests/astradscluster.yaml
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
```

```
~% kubectl apply -f manifests/astradscluster.yaml
```

응답:

```
astradscluster.astrads.netapp.io/cluster-multinodes-21209 configured
```

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

응답:

NAME	READY	STATUS	RESTARTS	AGE
astrads-cluster-controller...	1/1	Running	1	24h
astrads-deployment-support...	3/3	Running	0	24h
astrads-ds-cluster-multinodes-21209	1/1	Running		

```
~% 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

```
~% kubectl astrads clusters list
```

응답:

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

```
~% kubectl astrads drives list
```

응답:

DRIVE NAME	DRIVE ID	DRIVE STATUS	NODE NAME	CLUSTER NAME
scsi-36000..	c3e197f2...	Active	sti-rx2540...	cluster-multinodes-21209

## 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 드라이브 정보를 볼 수 있습니다.

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

```
$ kubectl get adscl -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.ctl.gdl.englab.netapp.com
  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 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 replace를 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

- ["kubectl 명령을 사용하여 Astra Data Store 미리 보기 자산을 관리합니다"](#)

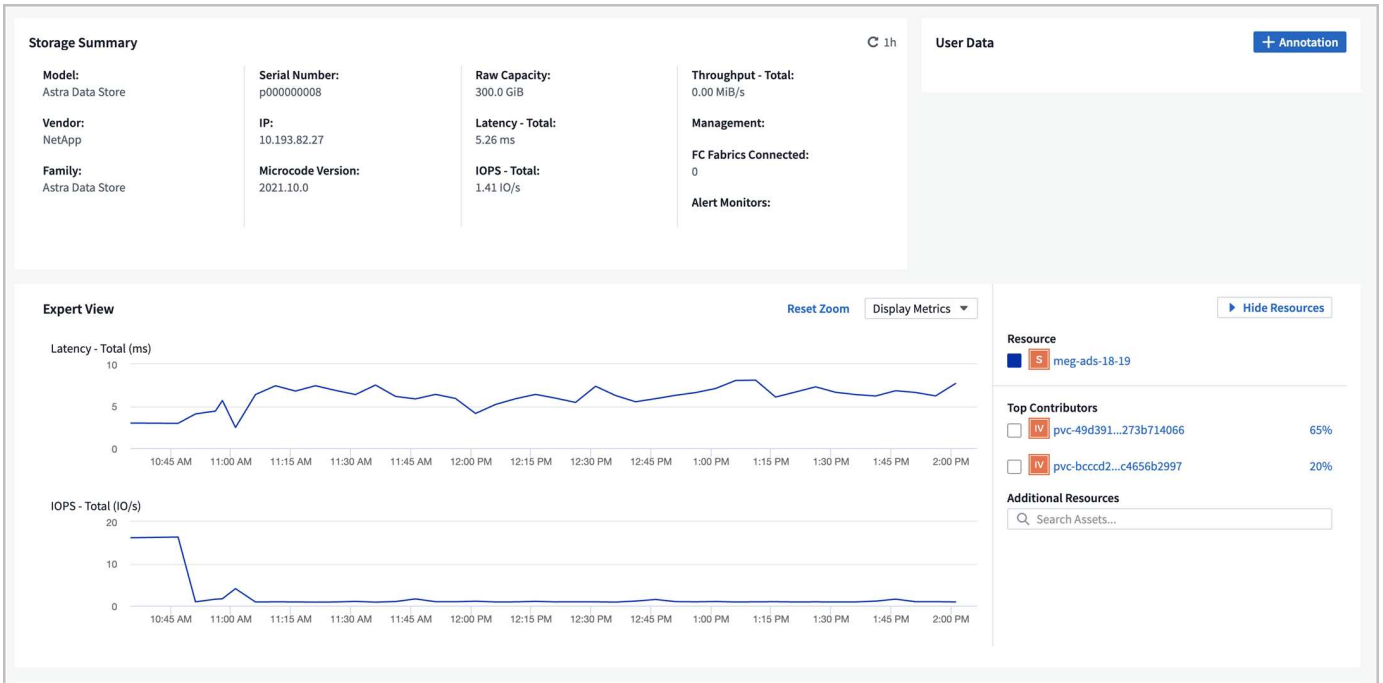
## Cloud Insights를 통해 메트릭을 모니터링합니다

Cloud Insights를 사용하여 Astra 데이터 저장소 미리 보기 메트릭을 모니터링할 수 있습니다.

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

다음은 Cloud Insights에 표시되는 몇 가지 Astra 데이터 저장소 미리 보기 메트릭입니다





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

## Cloud Insights 연결 필수 작업을 완료합니다

Astra 데이터 저장소를 Cloud Insights와 연결하기 전에 다음 사항을 수행해야 합니다.

- ["Astra Data Store Monitoring Operator를 설치합니다"](#) 이것은 Astra Data Store 미리 보기 설치 지침의 일부입니다.
- ["kubbeck-astrads 바이너리를 설치합니다"](#) 이것은 Astra Data Store 미리 보기 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 키 이 작업은 읽기/쓰기 작업, 획득 장치 설정 및 데이터 수집 프로세스 설정에 사용됩니다.
- \* Kubernetes API Server IP 주소 및 포트 \*. Astra Data Store 미리보기 클러스터를 모니터링하는 데 사용됩니다.
- \* Kubernetes API 토큰 \*. Kubernetes API를 호출하는 데 사용됩니다.
- \* 영구 볼륨 구성 \*. 영구 볼륨의 프로비저닝 방법에 대한 정보입니다. 자세한 내용은 아래의 "획득 장치"를 참조하십시오.

## 설치 스크립트를 다운로드하고 실행합니다

Cloud Insights는 모니터링 오퍼레이터를 통해 Astra 데이터 저장소 미리 보기 모니터링을 활성화하는 Bash 스크립트를 제공합니다. 설치 스크립트는 Astra Data Store Collector, Telegraf Agent 및 Fluent Bit Agent와 함께 획득 장치를 설치합니다.

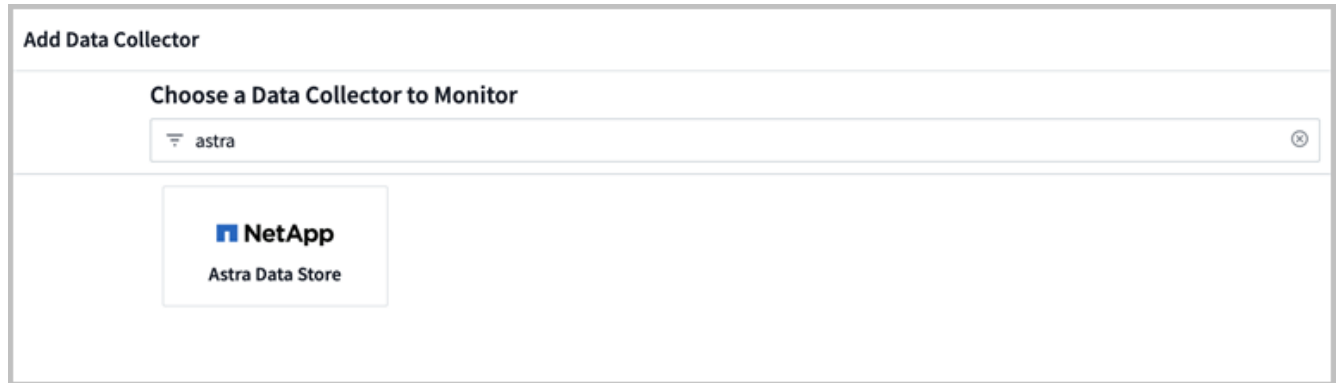
Cloud Insights 테넌트 도메인 이름과 선택한 API 액세스 키는 다운로드 시 설치 프로그램 스크립트에 포함됩니다.

그런 다음 다음과 같이 메트릭이 전송됩니다.

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


단계

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 API 액세스 토큰을 선택하거나 새 토큰을 생성합니다.
7. 지침에 따라 설치 프로그램 스크립트를 다운로드하고, 권한을 업데이트하고, 스크립트를 실행합니다.


이 스크립트에는 Cloud Insights 테넌트 URL 및 선택한 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-adsrads* 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 오류가 해결된 후 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. 용도와 도움말 텍스트를 보려면 `""-h""` 옵션을 사용하십시오.

설치 스크립트를 실행하면 다음과 같은 출력이 생성됩니다.

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Telegraf 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!
```

## 획득 장치 저장

획득 장치는 설치 파일, 구성 데이터 및 로그를 저장하기 위해 세 개의 영구 볼륨이 필요합니다. 모니터링 운영자는 기본 스토리지 클래스를 사용하여 영구 볼륨 청구를 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.

## 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
telegraf:
- name: ads-open-metric
  outputs:
  - sink: CI
  run-mode:
  - ReplicaSet
  substitutions:
  - key: URLS
    values:
    - http://astrads-metrics-service.astrads-
system.svc.cluster.local:9341
  - key: METRIC_TYPE
    value: ads-metric
  - key: ADS_CATEGORY
    value: netapp_ads
  - key: ADS_CLUSTER_NAME
    value: meg-ads-21-22-28-29-30
- name: agent
status:
  au-pod-status: UP
  au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9

```

## 설치 프로그램 스크립트 도움말입니다

설치 프로그램 스크립트에 대한 전체 도움말 텍스트가 아래에 나와 있습니다.

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

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 Key.
  -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.
```

## Cloud Insights 연결을 편집합니다

나중에 Kubernetes API 키 또는 Cloud Insights API 키를 편집할 수 있습니다.

- Kubernetes API 키를 업데이트하려면 Cloud Insights UI에서 Astra Data Store Collector를 편집해야 합니다.
- 원격 측정 및 로그에 사용되는 Cloud Insights API 키를 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.

### 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:

```
kubectl --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 API 키로 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 API 키를 사용하도록 Telegraf 및 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 작동기에서 획득 장치, 텔레그라프 및 Fluent 비트 구성을 제거할 수 있습니다.

### Astra Data Store 미리 보기 수집기를 제거합니다

1. Cloud Insights에 로그인합니다.
2. Admin \* > \* Data Collector \* 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 화면 오른쪽에서 케밥 메뉴를 선택하고 \* Delete \* 를 선택합니다.
5. 확인 페이지에서 \* 삭제 \* 를 클릭합니다.

획득 장치, 텔레그라프 및 **Fluent** 비트를 제거합니다

1. 상담원 CR 편집:

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

2. au 섹션을 찾아 isEnabled:false를 설정합니다
3. 유창한 비트 섹션을 찾아 "ads-tail-ci" 플러그인을 제거합니다. 플러그인이 더 이상 없으면 "fluent-bit" 섹션을 제거할 수 있습니다.
4. Telegraf 섹션을 찾아 "ads-open-metric" 플러그인을 제거합니다. 플러그인이 더 이상 없으면 Telegraf 섹션을 제거할 수 있습니다.

5. 출력 싱크 섹션을 찾아 이름이 "CI"인 싱크를 제거합니다.

6. 편집기 창을 저장하고 종료합니다.

모니터링 오퍼레이터는 Telegraf 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 AU가 실행 중인 노드에서 실제 디렉토리를 삭제합니다.

8. 획득 장치 포드가 삭제된 후 Cloud Insights에서 획득 장치를 삭제할 수 있습니다.

- Cloud Insights 메뉴에서 \* Admin \* > \* Data Collector \* 를 선택합니다.
- Acquisition Units(획득 단위) \* 탭을 클릭합니다.
- 획득 장치 포드 옆에 있는 메뉴를 클릭합니다.
- 삭제 \* 를 클릭합니다.

모니터링 작업자는 Telegraf 및 Fluent 비트 구성을 업데이트하고 획득 장치를 제거합니다.

## 메트릭 API 도움말을 엽니다

다음은 Astra Data Store 미리 보기에서 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
```

## 이벤트 로그를 구성하고 모니터링합니다

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

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

### Astra 데이터 저장소 미리 보기 클러스터 사용자 지정 리소스(CR)에서 모니터링 구성

Astra Data Store preview cluster 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: 이 에이전트는 모니터링 운영자용 CR입니다.
- '-r'은 선택 사항입니다.

### Cloud Insights를 설정합니다

을 참조하십시오 ["NetApp Cloud Insights 설정 방법"](#) Astra 데이터 저장소 미리 보기와 함께 사용할 수 있습니다.

### 이벤트 로그를 Elastic로 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.

## Astra Data Store 미리 보기와 함께 Astra Control Center를 사용합니다

Astra Control Center UI(사용자 인터페이스)를 사용하여 Astra Data Store 미리 보기 작업을 수행할 수 있습니다.

### Astra Data Store 미리 보기를 위한 Astra Control Center를 설정합니다

Astra Data Store 미리 보기를 위한 Astra Control Center UI를 사용하려면 다음 작업을 완료해야 합니다.

- 클러스터를 추가합니다. Astra Data Store 미리 보기가 설치된 기본 Kubernetes 클러스터입니다. 을 참조하십시오 ["Astra Data Store 미리 보기 클러스터를 Astra Control Center 설치로 가져옵니다"](#).
- 해당 클러스터에 대한 Astra Data Store 미리 보기 스토리지 백엔드를 추가합니다.



스토리지 백엔드를 추가하고 Astra Data Store 미리 보기가 있는 Kubernetes 클러스터가 없는 경우, 먼저 클러스터를 추가해야 합니다.

### Astra Control Center에서 수행할 수 있는 작업

Astra Data Store 미리 보기를 위한 Astra Control Center를 설정한 후 Astra Control Center UI를 사용하여 다음 작업을 수행할 수 있습니다.

- ["Astra Control Center를 사용하여 Astra Data Store Preview 자산의 상태를 모니터링합니다"](#).
- ["Astra Data Store 미리 보기 백엔드 스토리지를 관리합니다"](#).
- ["노드, 디스크 및 지속적인 PVC\(Volume Claim\)와 그 상태를 봅니다"](#).

### 를 참조하십시오

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

## Astra Data Store 미리 보기를 제거합니다

Astra Data Store 미리 보기를 제거하려면 워크로드, 바인딩, 볼륨, 익스포트 정책, 클러스터, 라이선스, 배포 환경 및 Astra Data Store 미리 보기 네임스페이스.

무엇을 '필요로 할거야

- 루트 관리 권한

Astra Data Store 미리 보기 제거 프로세스는 다음과 같은 고급 단계를 안내합니다.

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

## 기존 워크로드 및 바인딩을 제거합니다

Astra Data Store 미리 보기를 제거하기 전에 먼저 다음을 제거해야 합니다

- Astra Data Store 미리 보기를 스토리지 백엔드로 사용하는 모든 애플리케이션 워크로드
- Astra Data Store 미리 보기를 백엔드로 사용하는 Trident 바인딩

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

## Astra Data Store 미리 보기 클러스터와 컨트롤 플레인을 제거합니다

Astra Data Store 미리 보기를 수동으로 제거하려면 아래 단계를 따르십시오.

볼륨 및 익스포트 정책을 삭제합니다

클러스터를 삭제하기 전에 Astra Data Store 미리 보기 볼륨 및 익스포트 정책을 삭제해야 합니다.



먼저 볼륨을 삭제하고 정책을 내보내지 않으면 Astra Data Store 미리 보기 볼륨 개체가 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 미리보기 클러스터를 삭제합니다

클러스터를 삭제하면 클러스터 범위 리소스와 함께 Astra Data Store 미리보기 클러스터 객체 사용자 지정 리소스 (CR)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 미리 보기 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 Preview 라이선스를 삭제합니다.

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

## Astra Data Store 미리 보기 설치를 삭제합니다

클러스터에서 컨트롤러, 운영자, 네임스페이스 및 지원 포드를 삭제하십시오.

### 1. Astra Data Store 미리 보기 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 미리 보기 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 Store 미리보기를 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.

#### 3. 모든 Kubernetes 노드에서 `/var/lib/kubelet/config.yaml`을 제거합니다.



## 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
----	-----	--------	--------	---------	-------

# 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](https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra)

## 도움을 받으십시오

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



Astra Data Store 미리보기에 대한 커뮤니티 기술 지원을 받을 수 있습니다. 를 사용하여 케이스를 생성합니다 "NetApp Support 사이트(NSS)" 은(는) 미리보기 릴리스에 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 Store 미리 보기와 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](mailto: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 미리보기 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 미리 보기](#)" 또는 AutoSupport CR이 클러스터에 적용된 후입니다. 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: CR에 정의된 간격으로 ASUP 번들을 생성합니다.
- 사용자 트리거: 로그를 볼 수 있도록 사용자 고유의 ASP를 수동으로 생성할 수 있습니다.
- Coredumps: 노드에 코어 덤프가 있는 경우 ASUP가 생성되고 코어가 NetApp으로 전송되어 추가 조사가 수행됩니다.
- CallHome 이벤트 기반: 운영 체제에서 특정 callhome 이벤트를 통해 ASUP가 생성됩니다.
- Kubernetes 이벤트 기반: ASUP는 제어 플레인의 특정 Kubernetes 이벤트에서 생성됩니다.

이러한 트리거 시나리오는 다음 AutoSupport 유형 중 하나를 생성합니다.

- ControlPlane AutoSupport: Astra 데이터 저장소 미리 보기 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 설치 기본적으로 에 대해 정기적인 이벤트 보고를 사용하도록 설정되어 있습니다. "[Astra Data Store 클러스터 설치 미리 보기](#)". 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>": 이벤트 이름을 수집하는 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 미리 보기"](#). 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>": 이벤트 이름을 수집하는 이벤트 이름은 구성 요소에 미리 정의되어 있어야 합니다. YAML(지원 컨트롤러에 탑재됨).

성능 이벤트의 예:

```
kubectl-astrads 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 astrads 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 Store 미리 보기에서 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\(로그인 필요\)"](#)



# 법적 고지

""

""

"Astra Data Store 21.12 릴리스에 대한 고지 사항"

## Copyright Information

Copyright © 2022 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

## Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.