



시작하십시오 Astra Data Store

NetApp
June 17, 2022

This PDF was generated from <https://docs.netapp.com/ko-kr/astra-data-store/get-started/requirements.html> on June 17, 2022. Always check docs.netapp.com for the latest.

목차

시작하십시오	1
Astra 데이터 저장소 요구 사항	1
Astra Data Store를 빠르게 시작합니다.....	5
Astra Data Store를 설치합니다	7
Astra Data Store 구성 요소를 설정합니다	24
Astra Data Store 조기 액세스 프로그램(EAP) 릴리스 제한	35
Astra Data Store에 대한 질문과 대답	36

시작하십시오

Astra 데이터 저장소 요구 사항

먼저 귀사의 환경이 Astra Data Store 요구 사항을 충족하는지 확인하십시오.

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

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



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

Kubernetes 작업자 노드 리소스

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

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

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

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



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

각 작업자 노드에는 다음과 같은 추가 요구 사항이 있습니다.

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

하드웨어 및 소프트웨어

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

하드웨어 플랫폼

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



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

스토리지

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

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



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

소프트웨어

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

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



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

네트워킹

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

또한 다음 표에 설명된 대로 각 노드를 구성할 수 있습니다.



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

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



이러한 구성에는 VLAN 태그가 사용되지 않습니다.

방화벽 고려 사항

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

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

- MVIP: 하나의 클러스터 노드에 구성된 관리 가상 IP 주소입니다

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

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

아스트라 트리덴트

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

컨테이너 네트워크 인터페이스

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

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



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

라이센싱

Astra Data Store에는 전체 기능을 사용하려면 유효한 라이선스가 필요합니다.

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

다음 단계

를 봅니다 ["빠른 시작"](#) 개요.


를 참조하십시오

["Astra 데이터 저장소 제한"](#)

Astra Data Store를 빠르게 시작합니다


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

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

 **Kubernetes 클러스터 요구 사항을 검토합니다**


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

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

 **Astra Data Store를 다운로드하고 설치합니다**

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

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

 **일부 초기 설정 작업을 완료합니다**

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

에 대해 자세히 알아보십시오 ["초기 설정 프로세스"](#).

Astra Data Store 설정을 마치면 다음 단계를 수행할 수 있습니다.

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

["Astra Data Store를 설치합니다"](#).

Astra Data Store를 설치합니다

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

설치 옵션

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

필요한 것

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

Astra Data Store를 설치합니다

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

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



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



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

Astra Data Store 번들을 다운로드하고 이미지를 추출합니다

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



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

2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

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

3. 디렉토리 생성:

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

4. 이미지 추출:

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



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

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

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



kubbeck 명령을 사용하여 kubctl 바이너리가 설치된 경로를 찾습니다.

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

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



아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

- a. 레지스트리에 로그인합니다.

```
docker login [your_registry_path]
```

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

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

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

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

OpenShift 절차

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

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

예 1. 세부 정보

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

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

1. 네임스페이스 만들기:

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

샘플: ads_namespace.YAML

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

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

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

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

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

1. 사용자 지정 SCC 생성:

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

샘플: ads_privileged_csC.yAML

```

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

```

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

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

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

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

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

1. 정의된 역할 및 역할 바인딩을 생성합니다.

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

샘플: OC_ROLE_BINDINGS.YAML

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

```

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

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

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

```

개인 이미지 레지스트리를 구성합니다

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

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

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

2. 비밀 만들기:

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

3. 서비스 계정에 비밀 구성 정보 추가:

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



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

Astra Data Store 운영자를 설치합니다

1. Astra Data Store 매니페스트 나열:

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

응답:

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

2. kubectk apply를 사용하여 운영자를 배치한다.

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

응답:



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

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



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

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

```

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

```

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

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

응답:

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

Astra Data Store 버전 YAML을 배포하십시오

1. kubect apply를 이용한 구축:

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

2. Pod가 실행 중인지 확인합니다.

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

응답:

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

Astra Data Store 라이선스를 적용합니다

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

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

2. 라이선스가 추가되었는지 확인합니다.

```
kubectl astrads license list
```

응답:

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

Astra Data Store 클러스터를 설치합니다

1. YAML 파일을 엽니다.

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

2. YAML 파일에서 다음 값을 편집합니다.



YAML 파일의 간단한 예는 다음과 같습니다.

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



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

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

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

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



간결성을 위해 아래의 YAML 샘플에서 몇 가지 의견이 제거되었습니다.

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

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

3. "kubctl apply"를 사용하여 클러스터를 구축합니다.

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

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

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

샘플 반응:

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

5. 클러스터 배포 진행 상태 확인:

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

샘플 반응:

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

배포 관련 이벤트를 이해합니다

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

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

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

다음은 배포 중에 발생하는 주요 이벤트입니다.

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

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

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

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

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

Astra Control Center를 사용하여 Astra Data Store를 설치합니다

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

무엇을 '필요로 할거야

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

단계

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

다음 단계

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

Astra Data Store 구성 요소를 설정합니다

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



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

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

Astra Trident를 설치합니다

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

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



Trident 연산자는 수동으로 또는 Hrom을 사용하여 배포할 수 있습니다.

Kubernetes 스냅샷 CRD 및 컨트롤러를 설치합니다

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



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

무엇을 '필요로 할거야

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

단계

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

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

응답:

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

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

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

응답:

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

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

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

응답:

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

4. 설정 적용 - 스냅샷 - 컨트롤러.YAML:

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

응답:

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

5. RBAC-스냅샷-컨트롤러 적용.YAML:

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

응답:

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

6. CRD YAML 파일이 적용되었는지 확인합니다.

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

샘플 반응:

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

7. 스냅샷 컨트롤러 파일이 적용되었는지 확인합니다.

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

샘플 반응:

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

Astra Data Store를 스토리지 백엔드로 설정합니다

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

단계

1. 보안 터미널을 사용하여 ads_backend.json을 생성한다.

```
vi ads_backend.json
```

2. JSON 파일 구성:



JSON 샘플은 다음 단계를 따릅니다.

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

변환 예:

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

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

인코딩 예:

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

```

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

```

3. Trident 설치 프로그램을 다운로드한 디렉토리로 이동합니다.

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

4. 스토리지 백엔드를 생성합니다.

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

샘플 반응:

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

기본 **Astra Data Store** 스토리지 클래스를 생성합니다

Astra Trident 기본 스토리지 클래스를 생성하고 스토리지 백엔드에 적용합니다.

단계

1. 트리덴트 CSI 스토리지 클래스를 생성합니다.

a. ADS_SC_Example.YAML 생성:

```
vi ads_sc_example.yaml
```

예:


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

b. 트리덴트 CSI 생성:

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

응답:

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

2. 스토리지 클래스가 추가되었는지 확인합니다.

```
kubectl get storageclass
```

응답:

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

3. Trident 설치 프로그램을 다운로드한 디렉토리로 이동합니다.

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

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

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

샘플 반응:

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

```

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

```

```

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

```

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

Astra Data Store 모니터링을 구성합니다

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

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



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

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

단계

1. 다음 설치 명령을 실행합니다.

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

2. 모니터링을 위해 Astra Data Store 구성:

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

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

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

Astra Data Store 조기 액세스 프로그램(EAP) 릴리스 제한

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

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

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

Astra Data Store에 대한 질문과 대답

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

일반적인 질문

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

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

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

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

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

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

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

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

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

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

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

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

라이센싱

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

을 참조하십시오 ["Astra 데이터 저장소 요구 사항"](#).

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

Kubernetes 클러스터에 Astra Data Store 설치 및 사용

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

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

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

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

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

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

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

문제 해결 및 지원

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

"NetApp 컨테이너 여유 채널"

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

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

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

Astra Data Store 업그레이드 중

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

저작권 정보

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

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

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

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구입의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

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

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

상표 정보

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