시작하십시오
Astra Data Store

NetApp
July 19, 2022

목차

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	• 3(별도의 캐시 디바이스가 있는 경우) • 4(캐시 디바이스가 없는 경우)	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
  - SATA 또는 NVMe TLC SSD
- * VM 기반 구축 *: ESXi 클러스터에서 호스팅되는 Linux VM의 Kubernetes 클러스터에 설치된 Astra Data Store 미리보기
  - SATA, 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	• 노드당 2개: ◦ 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	• 노드당 3개: ◦ 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

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 미리보기를 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. 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 라이센스가 있어야 모든 기능을 사용할 수 있습니다. "여기에서 등록하십시오" Astra Data Store Preview 라이센스를 얻으려면 라이센스 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.

를 참조하십시오

"Astra Data Store 미리보기 제한"

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 라이센스를 사용할 수 있습니다.

"여기에서 등록하십시오" Astra Data Store Preview 라이센스를 얻으려면

span class="image"><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png" Alt="One"> 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".

span class="image"><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png" Alt="Two">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 라이센스를 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"><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png" Alt="3">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".

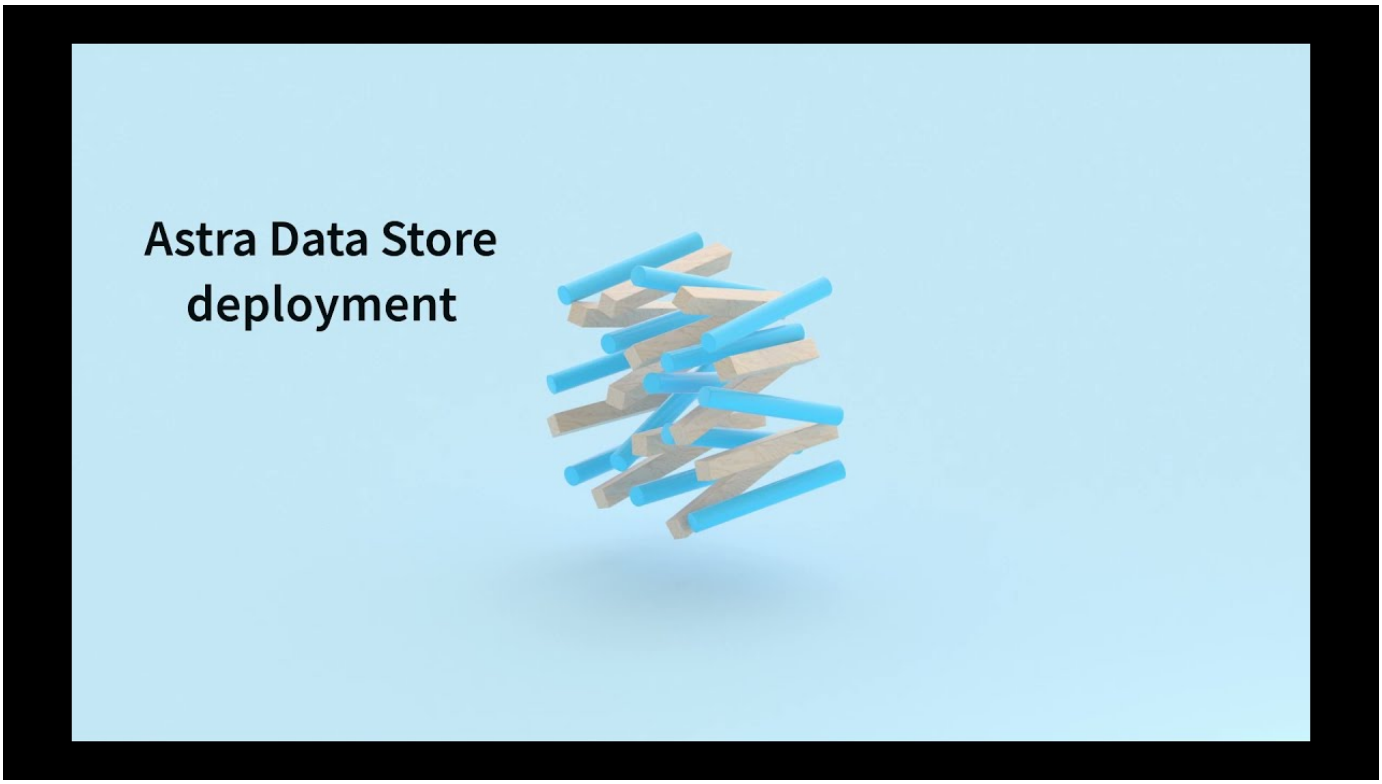
Astra Data Store 미리 보기 설정을 마치면 다음 작업을 수행할 수 있습니다.

- 유지보수 모드에 노드 배치, 드라이브 교체 또는 노드 교체 등의 작업을 포함하여 클러스터를 관리하려면 kubbtl 명령과 kebtl astrand 확장을 사용하십시오. 에 대해 자세히 알아보십시오 "Astra Data Store 미리 보기와 함께 kubeck 명령을 사용하는 방법".
- 모니터링 엔드포인트를 구성합니다. 에 대해 자세히 알아보십시오 "모니터링 엔드포인트 구성".

"Astra Data Store 미리 보기를 설치합니다".

Astra Data Store 미리 보기를 설치합니다

Astra Data Store 미리 보기를 설치하려면 에서 설치 번들을 다운로드하십시오 "NetApp Support 사이트" 이 절차에 설명된 설치 단계를 완료합니다.



무엇을 '필요로 할거야

- "설치를 시작하기 전에 Astra Data Store 미리 보기 구축을 위한 환경을 준비합니다".
- 에 액세스합니다 "NetApp Support 사이트". "등록" 전체 액세스 권한이 없는 NetApp Support 사이트 계정이 없는 경우 미리 보기를 위해
- A "NetApp 라이센스 파일(NLF)" Astra Data Store 미리 보기입니다. 라이센스 다운로드 지침이 사용자에게 전송됩니다 "가입하세요".
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubecononfig.
- 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]
- [OpenShift procedure]
- [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 미리 보기를 사용하여 비밀이 있는 이미지 등록부와 함께 작업하려면 을 참조하십시오 "이 KB입니다

".

Astra Data Store 미리보기 번들을 다운로드하고 이미지를 추출합니다

1. 에 로그인합니다 "NetApp Support 사이트" Astra Data Store Preview 번들('2021.12.01_ads.tar')을 다운로드하십시오.
2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify 2021.12_ads.pub -signature 2021.12_ads.sig
2021.12.01_ads.tar
```

3. 이미지 추출:

```
tar -xvf 2021.12.01_ads.tar
```

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

1. k8kubbeck 바이너리가 설치된 표준 경로(예: "/usr/bin/")로 이미지를 추출하기 위해 사용한 디렉토리에서 kubbctl-astrs 바이너리를 복사합니다. kubbctl-astrs는 Astra Data Store Preview 클러스터를 설치 및 관리하는 사용자 지정 kubtl 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
```

OpenShift 절차

다음 절차는 Red Hat OpenShift Container Platform(OCP)에서 배포용으로만 필요합니다. OCP가 아닌 Kubernetes 클러스터 구축에 대해서는 이 절차를 건너뛸 수 있습니다.

예 **1.** 세부 정보

모든 Astra Data Store 미리 보기 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 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.

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 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.

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
```

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:

ⓘ 네임스페이스 응답은 표준 설치를 수행했는지 OCP 설치를 수행했는지에 따라 다를 수 있습니다.

```
namespace/astrads-system 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/astradsdeployments.astrads
.netapp.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/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/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. kubeck을 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 라이센스를 적용합니다

1. 미리 보기에 등록할 때 얻은 NetApp 라이센스 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. 라이센스가 추가되었는지 확인합니다.

```
kubectl astrads license list
```

응답:

```
NAME        ADSCLUSTER                 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 라이센스를 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 쉼표로 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 '<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.

        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 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 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: "[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. "kubbtl apply"를 사용하여 클러스터를 구축합니다.

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

4. (OCP만 해당) SELinux가 설정된 경우 Astra Data Store preview 클러스터의 노드에서 다음 디렉토리에 대한 'elinux' 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-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-nzkkr 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
AGE

astrads-example-cluster   created   2021.10.0   p100000006
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.

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

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

모니터링 운전자를 설치합니다

(선택 사항) Astra Data Store Preview를 Astra Control Center로 가져오지 않는 경우 모니터링 운용자를 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 -n netapp-monitoring -r [YOUR REGISTRY] setup
```

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.snap
shot.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.sna
pshot.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
torage.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:

   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
   ```

   예:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
mountOptions:
  - vers=4.1
```

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.

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 합니까? * kubtl 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 라이센스가 필요합니까? * 예, Astra Data Store 미리 보기를 사용하려면 NetApp 라이센스 파일(NLF)이 필요합니다.

을 참조하십시오 "Astra 데이터 저장소 요구 사항".

- Astra 데이터 저장소 미리 보기 라이센스는 얼마나 유효합니까? * Astra 데이터 저장소 미리 보기 라이센스의 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 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" 및 "Astra Data Store 미리보기 제한".

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.