# 시작하십시오
## Astra Control Center

NetApp
November 21, 2023

This PDF was generated from https://docs.netapp.com/ko-kr/astra-control-center-2208/get-started/requirements.html on November 21, 2023. Always check docs.netapp.com for the latest.

# 목차

시작하십시오 . . . . . 1
Astra Control Center 요구 사항 . . . . . 1
Astra Control Center를 빠르게 시작합니다 . . . . . 7
설치 개요 . . . . . 8
Astra Control Center를 설정합니다 . . . . . 57
Astra Control Center에 대한 질문과 대답 . . . . . 76

# 시작하십시오

## Astra Control Center 요구 사항

먼저 운영 환경, 애플리케이션 클러스터, 애플리케이션, 라이센스 및 웹 브라우저의 준비 상태를 확인하십시오.

- 구현할 수 있습니다
- 지원되는 스토리지 백엔드
- 애플리케이션 클러스터 요구사항
- [설명합니다]
- 복제 사전 요구 사항
- 인터넷 접속
- [라이센스]
- 온프레미스 Kubernetes 클러스터의 수신
- 네트워킹 요구 사항
- 지원되는 웹 브라우저

### 구현할 수 있습니다

Astra Control Center는 다음과 같은 유형의 운영 환경에서 검증되었습니다.

- Google Anthos 1.10 또는 1.11
- Kubernetes 1.22 ~ 1.24
- RKE(Rancher Kubernetes Engine):
  - RKE 1.2.16 w/Rancher 2.5.12 및 RKE 1.3.3 w/2.6.3
  - RKE 2 (v1.23.6 + rke2r2), Rancher 2.6.3
- Red Hat OpenShift Container Platform 4.8, 4.9 또는 4.10
- VMware Tanzu Kubernetes Grid 1.4 또는 1.5
- VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2 또는 1.13

Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다. Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

| 구성 요소 | 요구 사항 |
|---|---|
| 스토리지 백엔드 용량입니다 | 최소 500GB가 제공됩니다 |
| 작업자 노드 | 최소 3개의 작업자 노드, 각각 4개의 CPU 코어, 12GB RAM |
| FQDN 주소입니다 | Astra Control Center의 FQDN 주소입니다 |

| 구성 요소 | 요구 사항 |
| --- | --- |
| 아스트라 트리덴트 | Astra Trident 21.10.1 이상 SnapMirror 기반 애플리케이션 복제를 위해 Astra Trident 22.07 이상 설치 및 구성되었습니다 |

이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

- * 이미지 레지스트리 *: Astra Control Center 빌드 이미지를 푸시할 수 있는 기존 개인 Docker 이미지 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다.
- * Astra Trident/ONTAP 구성 *: Astra Control Center를 사용하려면 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Astra Trident에서 제공하는 다음과 같은 ONTAP 드라이버를 지원합니다.
  - ONTAP - NAS
  - ONTAP-SAN
  - ONTAP-SAN - 경제성

OpenShift 환경에서 앱을 복제하는 동안, Astra Control Center는 OpenShift가 볼륨을 마운트하고 파일 소유권을 변경할 수 있도록 허용해야 합니다. 따라서 이러한 작업을 허용하려면 ONTAP 볼륨 내보내기 정책을 구성해야 합니다. 다음 명령을 사용하여 이 작업을 수행할 수 있습니다.

1. `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser sys`
2. `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -anon 65534`

두 번째 OpenShift 운영 환경을 관리되는 컴퓨팅 리소스로 추가할 계획이라면 Astra Trident Volume Snapshot 기능이 활성화되어 있는지 확인해야 합니다. Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트하려면 "공식 Astra Trident 지침을 참조하십시오".

**VMware Tanzu Kubernetes Grid** 클러스터 요구 사항

VMware Tanzu Kubernetes Grid(TKG) 또는 Tanzu Kubernetes Grid Integrated Edition(TKGi) 클러스터에서 Astra Control Center를 호스팅하는 경우 다음 사항을 고려하십시오.

- Astra Control에서 관리하려는 모든 애플리케이션 클러스터에서 TKG 또는 TKGi 기본 스토리지 클래스 적용을 비활성화합니다. 를 편집하여 이 작업을 수행할 수 있습니다 `TanzuKubernetesCluster` 리소스 를 확인하십시오.
- Astra Control Center 설치의 일부로 POD 보안 정책(PSP) 제한 환경에서 다음 리소스가 생성됩니다.
  - POD 보안 정책
  - RBAC 역할
  - RBAC RoleBinding RBAC 역할 및 RoleBinding 리소스가 에 만들어집니다 `netapp-acc` 네임스페이스.
- TKG 또는 TKGi 환경에 Astra Control Center를 구축하는 경우 Astra Trident에 대한 특정 요구 사항을 숙지하십시오. 자세한 내용은 를 참조하십시오 "Astra Trident 문서".

> 기본 VMware TKG 및 TKGi 구성 파일 토큰은 구축 후 10시간 후에 만료됩니다. Tanzu 포트폴리오 제품을 사용하는 경우, Astra Control Center와 관리되는 애플리케이션 클러스터 간의 연결 문제를 방지하기 위해 만료되지 않는 토큰이 포함된 Tanzu Kubernetes Cluster 구성 파일을 생성해야 합니다. 자세한 내용은 를 참조하십시오 "VMware NSX-T 데이터 센터 제품 설명서"

**Google Anthos** 클러스터 요구 사항

Google Anthos 클러스터에서 Astra Control Center를 호스팅할 때 Google Anthos에는 기본적으로 MetalLB 로드 밸런서와 Istio 수신 게이트웨이 서비스가 포함되어 있으므로 설치 중에 Astra Control Center의 일반적인 수신 기능을 사용할 수 있습니다. 을 참조하십시오 "Astra Control Center를 구성합니다" 를 참조하십시오.

## 지원되는 스토리지 백엔드

Astra Control Center는 다음과 같은 스토리지 백엔드를 지원합니다.

- NetApp ONTAP 9.5 이상 AFF 및 FAS 시스템
- NetApp ONTAP 9.8 이상 SnapMirror 기반 애플리케이션 복제를 위한 AFF 및 FAS 시스템
- NetApp Cloud Volumes ONTAP를 참조하십시오

Astra Control Center를 사용하려면 수행해야 할 작업에 따라 다음과 같은 ONTAP 라이센스가 있는지 확인합니다.

- 플랙스클론
- SnapMirror: 선택 사항. SnapMirror 기술을 사용하여 원격 시스템에 복제하는 경우에만 필요합니다. 을 참조하십시오 "SnapMirror 라이센스 정보".
- S3 라이센스: 선택 사항. ONTAP S3 버킷에만 필요

ONTAP 시스템에 필요한 라이센스가 있는지 확인할 수 있습니다. 을 참조하십시오 "ONTAP 라이센스 관리".

## 애플리케이션 클러스터 요구사항

Astra Control Center에는 Astra Control Center에서 관리하려는 클러스터에 대한 다음과 같은 요구 사항이 있습니다. 이러한 요구 사항은 관리하려는 클러스터가 Astra Control Center를 호스팅하는 운영 환경 클러스터인 경우에도 적용됩니다.

- Kubernetes의 최신 버전입니다 "스냅샷 컨트롤러 구성 요소입니다" 이(가) 설치되었습니다
- Astra Trident "볼륨스냅샷 클래스 개체" 관리자가 정의했습니다
- 클러스터에 기본 Kubernetes 스토리지 클래스가 있습니다
- Astra Trident를 사용하도록 스토리지 클래스를 하나 이상 구성했습니다

> 애플리케이션 클러스터에는 가 있어야 합니다 `kubeconfig.yaml` 한 _CONTEXT_ELEMENT만 정의하는 파일입니다. 에 대한 Kubernetes 설명서를 참조하십시오 "kubecononfig 파일 생성에 대한 정보입니다".

> Rancher 환경에서 애플리케이션 클러스터를 관리할 때 에서 애플리케이션 클러스터의 기본 컨텍스트를 수정합니다 `kubeconfig` Rancher API 서버 컨텍스트 대신 컨트롤 플레인 컨텍스트를 사용하기 위해 Rancher에서 제공하는 파일입니다. 따라서 Rancher API 서버의 부하가 줄어들고 성능이 향상됩니다.

## 설명합니다

Astra Control에는 다음과 같은 애플리케이션 관리 요구 사항이 있습니다.

- * 라이선스 *: Astra Control Center를 사용하여 애플리케이션을 관리하려면 Astra Control Center 라이센스가 필요합니다.
- * Namespaces *: Astra Control은 앱이 단일 네임스페이스 이상의 범위를 포괄하지 않도록 하지만 네임스페이스에는 여러 개의 앱이 포함될 수 있습니다.
- * StorageClass *: StorageClass가 명시적으로 설정된 애플리케이션을 설치하고 앱을 복제해야 하는 경우 클론 작업의 타겟 클러스터에 원래 지정된 StorageClass가 있어야 합니다. 명시적으로 StorageClass를 동일한 StorageClass가 없는 클러스터로 설정한 애플리케이션을 클론 복제하면 실패합니다.
- * Kubernetes 리소스 *: Astra Control에서 수집하지 않은 Kubernetes 리소스를 사용하는 애플리케이션에는 전체 앱 데이터 관리 기능이 없을 수 있습니다. Astra Control은 다음과 같은 Kubernetes 리소스를 수집합니다.

| | | |
|---|---|---|
| 클러스터 역할 | ClusterRoleBinding 을 참조하십시오 | ConfigMap을 클릭합니다 |
| 경작업 | 사용자 지정 리소스 정의 | CustomResource 를 선택합니다 |
| DemonSet | DeploymentConfig(배포 구성 | HorizontalPodAutoscaler |
| 침투 | 뮤atingWebhook | 네트워크 정책 |
| PersistentVolumeClaim | 포드 | 팟캐스트 예산 |
| 팟캐스트 템플릿 | ReplicaSet입니다 | 역할 |
| RoleBinding 을 클릭합니다 | 루트 | 비밀 |
| 서비스 | 서비스 계정 | StatefulSet 을 선택합니다 |
| Webhook을 확인합니다 | | |

## 복제 사전 요구 사항

Astra Control 애플리케이션 복제를 시작하려면 먼저 다음과 같은 사전 요구 사항을 충족해야 합니다.

- 원활한 재해 복구를 위해 세 번째 장애 도메인 또는 보조 사이트에 Astra Control Center를 배포하는 것이 좋습니다.
- 앱의 호스트 Kubernetes 클러스터 및 대상 Kubernetes 클러스터를 사용할 수 있고 서로 다른 장애 도메인 또는 사이트에서 이상적인 두 ONTAP 클러스터에 연결할 수 있어야 합니다.
- ONTAP 클러스터와 호스트 SVM이 페어링되어야 합니다. 을 참조하십시오 "클러스터 및 SVM 피어링 개요".
- 타겟 클러스터의 Trident에서 페어링된 원격 SVM을 사용할 수 있어야 합니다.
- Trident 버전 22.07 이상이 소스 및 대상 ONTAP 클러스터 모두에 있어야 합니다.
- 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기식 라이센스를 설정해야 합니다. 을 참조하십시오 "ONTAP의 SnapMirror 라이센스 개요".
- ONTAP 스토리지 백엔드를 Astra 제어 센터에 추가할 때 액세스 방법이 있는 "admin" 역할을 사용하여 사용자 자격 증명을 적용합니다 `http` 및 `ontapi` 두 ONTAP 클러스터에서 모두 활성화 을 참조하십시오 "사용자 계정 관리" 를 참조하십시오.
- 소스 및 대상 Kubernetes 클러스터와 ONTAP 클러스터는 Astra Control에서 관리해야 합니다.

> 다른 클러스터 또는 사이트에서 실행 중인 다른 앱을 반대 방향으로 동시에 복제할 수 있습니다. 예를 들어, 애플리케이션 A, B, C를 데이터 센터 1에서 데이터 센터 2로 복제하고 애플리케이션 X, Y, Z를 데이터 센터 2에서 데이터 센터 1로 복제할 수 있습니다.

자세한 내용을 알아보십시오 "SnapMirror 기술을 사용하여 원격 시스템에 애플리케이션을 복제합니다".

## 지원되는 응용 프로그램 설치 방법

Astra Control은 다음과 같은 응용 프로그램 설치 방법을 지원합니다.

- * 매니페스트 파일 *: Astra Control은 kubctl을 사용하여 매니페스트 파일에서 설치된 앱을 지원합니다. 예를 들면 다음과 같습니다.

```
kubectl apply -f myapp.yaml
```

- * Helm 3 *: Helm을 사용하여 앱을 설치하는 경우 Astra Control에 Helm 버전 3이 필요합니다. Helm 3(또는 Helm 2에서 Helm 3으로 업그레이드)과 함께 설치된 앱의 관리 및 클론 생성이 완벽하게 지원됩니다. Helm 2가 설치된 앱 관리는 지원되지 않습니다.
- * 운용자 구축 앱 *: Astra Control은 네임스페이스 범위 연산자와 함께 설치된 앱을 지원합니다. 다음은 이 설치 모델에 대해 검증된 몇 가지 응용 프로그램들입니다.
  - "아파치 K8ssandra"
  - "젠킨스 CI"
  - "Percona XtraDB 클러스터"

> 운영자와 설치하는 앱은 동일한 네임스페이스를 사용해야 합니다. 운영자가 배포 .YAML 파일을 수정해야 할 수도 있습니다.

## 인터넷 접속

인터넷에 대한 외부 액세스 권한이 있는지 확인해야 합니다. 그렇지 않으면 NetApp Cloud Insights에서 모니터링 및 메트릭 데이터를 수신하거나 지원 번들을 보내는 등 일부 기능이 제한될 수 있습니다 "NetApp Support 사이트".

## 라이센스

Astra Control Center의 모든 기능을 사용하려면 Astra Control Center 라이센스가 필요합니다. NetApp에서 평가판 라이센스 또는 전체 라이센스를 받으십시오. 애플리케이션과 데이터를 보호하려면 라이센스가 필요합니다. 을 참조하십시오 "Astra Control Center의 특징" 를 참조하십시오.

Astra Control Center에 평가판 라이센스를 사용하면 라이센스를 다운로드한 날짜로부터 90일 동안 Astra Control Center를 사용할 수 있습니다. 등록하면 무료 평가판을 사용할 수 있습니다 "여기".

ONTAP 스토리지 백엔드에 필요한 라이센스에 대한 자세한 내용은 을 참조하십시오 "지원되는 스토리지 백엔드".

라이센스 작동 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

## 온프레미스 **Kubernetes** 클러스터의 수신

네트워크 수신 Astra Control Center 사용 유형을 선택할 수 있습니다. 기본적으로 Astra Control Center는 클러스터 차원의 리소스로 Astra Control Center 게이트웨이(서비스/traefik)를 배포합니다. 또한 Astra Control Center는 서비스 로드 밸런서가 사용자 환경에서 허용되는 경우 이를 사용할 수 있도록 지원합니다. 서비스 로드 밸런서를 사용하고 아직 서비스 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 자동으로 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다.

> Tanzu Kubernetes Grid 클러스터에 Astra Control Center를 호스팅하는 경우 를 사용하십시오 `kubectl get nsxlbmonitors -A` 수신 트래픽을 허용하도록 서비스 모니터가 이미 구성되어 있는지 확인하는 명령입니다. 기존 서비스 모니터가 새 로드 밸런서 구성을 무시하므로 MetalLB를 설치하면 안 됩니다.

자세한 내용은 을 참조하십시오 "부하 분산을 위한 수신 설정".

## 네트워킹 요구 사항

Astra Control Center를 호스팅하는 운영 환경은 다음 TCP 포트를 사용하여 통신합니다. 이러한 포트가 모든 방화벽을 통해 허용되는지 확인하고 Astra 네트워크에서 발생하는 HTTPS 송신 트래픽을 허용하도록 방화벽을 구성해야 합니다. 일부 포트에는 Astra Control Center를 호스팅하는 환경과 각 관리 클러스터(해당되는 경우) 간의 연결이 모두 필요합니다.

> Astra Control Center를 이중 스택 Kubernetes 클러스터에 구축할 수 있으며, Astra Control Center는 이중 스택 작업을 위해 구성된 애플리케이션 및 스토리지 백엔드를 관리할 수 있습니다. 이중 스택 클러스터 요구사항에 대한 자세한 내용은 를 참조하십시오 "Kubernetes 문서".

| 출처 | 목적지 | 포트 | 프로토콜 | 목적 |
|---|---|---|---|---|
| 클라이언트 PC | Astra 제어 센터 | 443 | HTTPS | UI/API 액세스 - Astra Control Center를 호스팅하는 클러스터와 관리되는 각 클러스터 간에 이 포트가 열려 있는지 확인합니다 |
| 소비자 평가 기준 | Astra Control Center 작업자 노드 | 9090 | HTTPS | 메트릭 데이터 통신 - 각 관리 클러스터가 Astra Control Center를 호스팅하는 클러스터의 이 포트에 액세스할 수 있는지 확인합니다 (양방향 통신 필요) |
| Astra 제어 센터 | Hosted Cloud Insights 서비스 | 443 | HTTPS | Cloud Insights 통신 |
| Astra 제어 센터 | Amazon S3 스토리지 버킷 공급자 | 443 | HTTPS | Amazon S3 스토리지 통신 |

| 출처 | 목적지 | 포트 | 프로토콜 | 목적 |
|---|---|---|---|---|
| Astra 제어 센터 | NetApp AutoSupport를 참조하십시오 | 443 | HTTPS | NetApp AutoSupport 커뮤니케이션 |

### 지원되는 웹 브라우저

Astra Control Center는 1280 x 720의 최소 해상도로 최신 버전의 Firefox, Safari 및 Chrome을 지원합니다.

### 다음 단계

를 봅니다 "빠른 시작" 개요.

# Astra Control Center를 빠르게 시작합니다

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

사용해 보세요! Astra Control Center를 사용해 보고 싶은 경우 90일 평가판 라이센스를 사용할 수 있습니다. 을 참조하십시오 "라이센스 정보" 를 참조하십시오.

1 **Kubernetes** 클러스터 요구사항을 검토하십시오

- Astra는 Trident에서 구성한 ONTAP 스토리지 백엔드 또는 Astra Data Store 스토리지 백엔드를 통해 Kubernetes 클러스터와 함께 작동합니다.
- 클러스터는 정상 상태에서 실행되어야 하며 3개 이상의 온라인 작업자 노드가 있어야 합니다.
- 클러스터가 Kubernetes를 실행 중이어야 합니다.

에 대해 자세히 알아보십시오 "Astra Control Center 요구 사항".

2 **Astra Control Center**를 다운로드하여 설치합니다

- 에서 Astra Control Center를 다운로드합니다 "NetApp Support 사이트 Astra Control Center 다운로드 페이지".
- 현지 환경에 Astra Control Center를 설치합니다.

  필요한 경우 Red Hat OperatorHub를 사용하여 Astra Control Center를 설치합니다.

  필요한 경우 Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치합니다.

에 대해 자세히 알아보십시오 "Astra Control Center 설치".

3 몇 가지 초기 설정 작업을 완료합니다

- Astra Control 라이센스와 지원 ONTAP 라이센스를 추가합니다.
- Kubernetes 클러스터 추가 및 Astra Control Center에서 세부 정보를 검색합니다.

- ONTAP 스토리지 백엔드를 추가합니다.
- 필요에 따라 앱 백업을 저장할 오브젝트 저장소 버킷을 추가합니다.

에 대해 자세히 알아보십시오 "초기 설정 프로세스".

4 **Astra Control Center**를 사용합니다

Astra Control Center 설정을 마치면 다음 단계를 수행하십시오.

- 앱을 관리합니다. 에 대해 자세히 알아보십시오 "앱 관리".
- 앱에 대한 보호 정책을 구성하고, 앱을 원격 시스템으로 복제하고, 앱을 복제 및 마이그레이션하여 앱을 보호합니다. 에 대해 자세히 알아보십시오 "앱 보호".
- 계정 관리(사용자, 역할, 사용자 인증을 위한 LDAP, 자격 증명, 리포지토리 연결 등) 에 대해 자세히 알아보십시오 "사용자 관리".
- 필요한 경우 NetApp Cloud Insights에 연결하여 Astra Control Center UI 내에서 시스템 상태, 용량 및 처리량에 대한 메트릭을 표시할 수 있습니다. 에 대해 자세히 알아보십시오 "Cloud Insights에 연결 중입니다".

5 이 빠른 시작에서 계속합니다

"Astra Control Center를 설치합니다".

### 자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"

## 설치 개요

다음 Astra Control Center 설치 절차 중 하나를 선택하여 완료합니다.

- "표준 프로세스를 사용하여 Astra Control Center를 설치합니다"
- "(Red Hat OpenShift를 사용하는 경우) OpenShift OperatorHub를 사용하여 Astra Control Center를 설치합니다"
- "Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치합니다"

### 표준 프로세스를 사용하여 **Astra Control Center**를 설치합니다

Astra Control Center를 설치하려면 NetApp Support 사이트에서 설치 번들을 다운로드하고 다음 단계를 수행하여 해당 환경에 Astra Control Center Operator and Astra Control Center를 설치합니다. 이 절차를 사용하여 인터넷에 연결되었거나 공기가 연결된 환경에 Astra Control Center를 설치할 수 있습니다.

Red Hat OpenShift 환경에서는 을 사용할 수 있습니다 "대체 절차" OpenShift OperatorHub를 사용하여 Astra Control Center를 설치하려면 다음을 수행합니다.

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".

- 사용자 환경에서 POD 보안 정책을 구성했거나 구성하려는 경우 POD 보안 정책 및 해당 정책이 Astra Control Center 설치에 어떤 영향을 미치는지 숙지하십시오. 을 참조하십시오 "POD 보안 정책 제한 사항 이해".
- 모든 클러스터 운영자가 양호한 상태이며 사용 가능한지 확인합니다.

```
kubectl get clusteroperators
```

- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

```
kubectl get apiservices
```

- 사용하려는 Astra FQDN이 이 클러스터에 라우팅될 수 있는지 확인합니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.
- 클러스터에 인증서 관리자가 이미 있는 경우 일부를 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하지 않습니다.

이 작업에 대해

Astra Control Center 설치 프로세스는 다음을 수행합니다.

- 에 Astra 구성 요소를 설치합니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다.
- 기본 계정을 만듭니다.
- 기본 관리 사용자 이메일 주소와 기본 1회 암호를 설정합니다. 이 사용자에게는 시스템에서 UI에 처음 로그인하는 데 필요한 소유자 역할이 할당됩니다.
- 모든 Astra Control Center Pod가 실행 중인지 확인하는 데 도움이 됩니다.
- Astra UI를 설치합니다.

> (Astra Data Store Early Access Program(EAP) 릴리즈에만 적용) Astra Control Center를 사용하여 Astra Data Store를 관리하고 VMware 워크플로우를 활성화하려면 에 Astra Control Center만 배포하십시오 `pcloud` 에 없는 네임스페이스입니다 `netapp-acc` 네임스페이스 또는 사용자 지정 네임스페이스는 이 절차의 단계에 설명되어 있습니다.

> 모든 Astra Control Center Pod를 삭제하지 않도록 설치 프로세스 내내 다음 명령을 실행하지 마십시오. `kubectl delete -f astra_control_center_operator_deploy.yaml`

> Docker Engine 대신 Red Hat의 Podman 명령을 사용하는 경우 Docker 명령 대신 Podman 명령을 사용할 수 있습니다.

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center 번들을 다운로드하고 포장을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다

- 인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다
- Astra Control Center 운영자를 설치합니다
- Astra Control Center를 구성합니다
- Astra 제어 센터 및 운전자 설치를 완료합니다
- 시스템 상태를 확인합니다
- 부하 분산을 위한 수신 설정
- Astra Control Center UI에 로그인합니다

**Astra Control Center** 번들을 다운로드하고 포장을 풉니다

1. Astra Control Center 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "NetApp Support 사이트".
2. 에서 Astra Control Center 인증서 및 키의 지퍼를 다운로드합니다 "NetApp Support 사이트".
3. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify AstraControlCenter-public.pub -signature
astra-control-center-[version].tar.gz.sig astra-control-center-
[version].tar.gz
```

4. 이미지 추출:

```
tar -vxzf astra-control-center-[version].tar.gz
```

**NetApp Astra kubtl** 플러그인을 설치합니다

NetApp 아스트라 `kubectl` 명령줄 플러그인을 사용하면 Astra Control Center 배포 및 업그레이드와 관련된 일반적인 작업을 수행할 때 시간을 절약할 수 있습니다.

필요한 것

NetApp은 다양한 CPU 아키텍처 및 운영 체제용 플러그인의 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다. Linux 및 Mac 운영 체제에서는 를 사용할 수 있습니다 `uname -a` 명령을 사용하여 이 정보를 수집합니다.

단계

1. 사용 가능한 NetApp Astra를 나열하십시오 `kubectl` 플러그인 바이너리를 만들고 운영 체제 및 CPU 아키텍처에 필요한 파일 이름을 적어 둡니다.

```
ls kubectl-astra/
```

2. 파일을 규격과 같은 위치에 복사합니다 `kubectl` 유틸리티. 이 예에서 는 입니다 `kubectl` 유틸리티는 에 있습니다 `/usr/local/bin` 디렉토리. 대치 `<binary-name>` 필요한 파일 이름:

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

이미지를 로컬 레지스트리에 추가합니다

1. 용기 엔진에 적합한 단계 시퀀스를 완료합니다.

**Docker** 를 참조하십시오

1. Astra 디렉토리로 이동합니다.

```
cd acc
```

2. Astra Control Center 이미지 디렉토리에 있는 패키지 이미지를 로컬 레지스트리로 푸시합니다. 명령을 실행하기 전에 다음 대체 작업을 수행합니다.
   - Bundle_file을 Astra Control 번들 파일 이름으로 바꿉니다(예: `acc.manifest.yaml`)를 클릭합니다.
   - my_registry를 Docker 리포지토리의 URL로 바꿉니다.
   - my_registry_user를 사용자 이름으로 바꿉니다.
   - my_registry_token을 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m BUNDLE_FILE -r MY_REGISTRY
-u MY_REGISTRY_USER -p MY_REGISTRY_TOKEN
```

팟맨

1. 레지스트리에 로그인합니다.

```
podman login [your_registry_path]
```

2. 설명에 명시된 대로 <your_registry> 대체를 만들어 다음 스크립트를 실행합니다.

```
# You need to be at the root of the tarball.
# You should see these files to confirm correct location:
#   acc.manifest.yaml
#   acc/

# Replace <YOUR_REGISTRY> with your own registry (e.g
registry.customer.com or registry.customer.com/testing, etc..)
export REGISTRY=<YOUR_REGISTRY>
export PACKAGENAME=acc
export PACKAGEVERSION=22.08.1-26
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
  # Load to local cache
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')

  # Remove path and keep imageName.
  astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')

  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/netapp/astra/${PACKAGENAME}
/${PACKAGEVERSION}/${astraImageNoPath}

  # Push to the local repo.
  podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. Astra Control Center 호스트 클러스터에 대한 KUBECONFIG를 내보냅니다.

```
export KUBECONFIG=[file path]
```

2. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

   a. 를 생성합니다 `netapp-acc-operator` 네임스페이스:

```
kubectl create ns netapp-acc-operator
```

   응답:

```
namespace/netapp-acc-operator created
```

b. 에 대한 암호를 만듭니다 `netapp-acc-operator` 네임스페이스. Docker 정보를 추가하고 다음 명령을 실행합니다.

> 자리 표시자입니다 `your_registry_path` 이전에 업로드한 이미지의 위치와 일치해야 합니다(예: `[Registry_URL]/netapp/astra/astracc/22.08.1-26`)를 클릭합니다.

```
kubectl create secret docker-registry astra-registry-cred -n netapp-
acc-operator --docker-server=[your_registry_path] --docker-username
=[username] --docker-password=[token]
```

샘플 반응:

```
secret/astra-registry-cred created
```

> 암호를 생성한 후 네임스페이스를 삭제하는 경우 네임스페이스를 다시 만든 후 네임스페이스에 대한 암호를 다시 생성해야 합니다.

c. 를 생성합니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다.

```
kubectl create ns [netapp-acc or custom namespace]
```

샘플 반응:

```
namespace/netapp-acc created
```

d. 에 대한 암호를 만듭니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다. Docker 정보를 추가하고 다음 명령을 실행합니다.

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-
acc or custom namespace] --docker-server=[your_registry_path]
--docker-username=[username] --docker-password=[token]
```

응답

```
secret/astra-registry-cred created
```

a. [[substep_kubecononfig_secret] (선택 사항) 설치 후 Astra Control Center에서 클러스터를 자동으로

관리하려는 경우 이 명령을 사용하여 배포할 Astra Control Center 네임스페이스 내에서 kubecononfig를 암호로 제공해야 합니다.

```
kubectl create secret generic [acc-kubeconfig-cred or custom secret name] --from-file=<path-to-your-kubeconfig> -n [netapp-acc or custom namespace]
```

**Astra Control Center** 운영자를 설치합니다

1. 디렉토리를 변경합니다.

```
cd manifests
```

2. Astra Control Center 운영자 배포 YAML을 편집합니다 (`astra_control_center_operator_deploy.yaml`)를 클릭하여 로컬 레지스트리 및 암호를 참조합니다.

```
vim astra_control_center_operator_deploy.yaml
```

> YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

a. 인증이 필요한 레지스트리를 사용하는 경우 의 기본 줄을 바꿉니다 `imagePullSecrets: []` 다음 포함:

```
imagePullSecrets:
- name: <astra-registry-cred>
```

b. 변경 `[your_registry_path]` 의 경우 `kube-rbac-proxy` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

c. 변경 `[your_registry_path]` 의 경우 `acc-operator-controller-manager` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

d. (Astra Data Store Preview를 사용하여 설치하는 경우) 와 관련된 알려진 문제를 참조하십시오 "스토리지 클래스 프로비저닝 및 YAML에 대한 추가 변경 사항".

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

3. Astra Control Center 운영자를 설치합니다.

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

샘플 반응:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

4. Pod가 실행 중인지 확인합니다.

```
kubectl get pods -n netapp-acc-operator
```

### **Astra Control Center**를 구성합니다

1. Astra Control Center 사용자 정의 리소스(CR) 파일을 편집합니다 (`astra_control_center_min.yaml`) 계정, AutoSupport, 레지스트리 및 기타 필요한 구성을 만들려면:

> `astra_control_center_min.yaml` 기본 CR이며 대부분의 설치에 적합합니다. 모든 것을 숙지합니다 "CR 옵션 및 잠재적 가치" 고객의 환경에 맞게 Astra Control Center를 올바르게 구축할 수 있습니다. 사용자 환경에 추가 사용자 지정이 필요한 경우 를 사용할 수 있습니다 `astra_control_center.yaml` 대체 CR입니다.

```
vim astra_control_center_min.yaml
```

> 인증이 필요하지 않은 레지스트리를 사용하는 경우 을 삭제해야 합니다 `secret` 줄 내부 `imageRegistry` 그렇지 않으면 설치가 실패합니다.

a. 변경 `[your_registry_path]` 이전 단계에서 이미지를 푸시한 레지스트리 경로로 이동합니다.

b. 를 변경합니다 `accountName` 계정에 연결할 이름에 대한 문자열입니다.

c. 를 변경합니다 `astraAddress` 브라우저에서 Astra에 액세스하기 위해 사용할 FQDN에 대한 문자열입니다.

사용하지 마십시오 `http://` 또는 `https://` 를 입력합니다. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.

d. 를 변경합니다 `email` 문자열을 기본 초기 관리자 주소로 설정합니다. 에서 사용할 이 이메일 주소를 복사합니다 나중에.

e. 변경 `enrolled` 을 눌러 AutoSupport to로 이동합니다 `false` 인터넷 연결이 없거나 보관되지 않은 사이트의 경우 `true` 연결된 사이트의 경우.

f. 외부 인증서 관리자를 사용하는 경우 에 다음 행을 추가합니다 `spec`:

```
spec:
  crds:
    externalCertManager: true
```

g. (선택 사항) 이름을 추가합니다 `firstName` 성을 입력합니다 `lastName` 계정에 연결된 사용자의 입니다. UI 내에서 이 단계를 지금 또는 나중에 수행할 수 있습니다.

h. (선택 사항) 을 변경합니다 `storageClass` 설치에 필요한 경우 다른 Trident storageClass 리소스에 대한 값입니다.

i. (선택 사항) 설치 후 클러스터를 Astra Control Center에서 자동으로 관리하려는 경우 이 클러스터에 kubecon무화과 같은 암호를 만들었습니다, 라는 이 YAML 파일에 새 필드를 추가하여 비밀의 이름을 입력합니다 `astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"`

j. 다음 단계 중 하나를 수행합니다.

- * 기타 수신 컨트롤러(ingressType: Generic) *: Astra Control Center의 기본 동작입니다. Astra Control Center를 배포한 후 URL을 사용하여 Astra Control Center를 노출하도록 수신 컨트롤러를 구성해야 합니다.

  기본 Astra Control Center 설치는 게이트웨이를 설정합니다 (`service/traefik`)를 입력합니다 `ClusterIP`. 이 기본 설치에서는 트래픽을 이 컨트롤러로 라우팅하기 위해 추가적으로 Kubernetes IngPressController/Ingress를 설정해야 합니다. 침투를 사용하려면 를 참조하십시오 "부하 분산을 위한 수신 설정".

- * 서비스 로드 밸런서(ingressType:AccTraefik) *: IngressController를 설치하거나 수신 리소스를 생성하지 않으려면 를 설정합니다 `ingressType` 를 선택합니다 `AccTraefik`.

  그러면 Astra Control Center가 구축됩니다 `traefik` Kubernetes 로드 밸런서 유형 서비스로서의 게이트웨이

  Astra Control Center는 "loadbalancer" 유형의 서비스를 사용합니다. (`svc/traefik` Astra Control Center 네임스페이스에서), 액세스 가능한 외부 IP 주소를 할당해야 합니다. 로드 밸런서가 사용자 환경에서 허용되고 아직 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 또는 다른 외부 서비스 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다.

  ⓘ "로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을 참조하십시오 "요구 사항".

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

**Astra** 제어 센터 및 운전자 설치를 완료합니다

1. 이전 단계에서 아직 작성하지 않은 경우 를 만듭니다 `netapp-acc` (또는 사용자 지정) 네임스페이스:

   ```
   kubectl create ns [netapp-acc or custom namespace]
   ```

   샘플 반응:

   ```
   namespace/netapp-acc created
   ```

2. 에 Astra Control Center를 설치합니다 `netapp-acc` (또는 사용자 지정) 네임스페이스:

   ```
   kubectl apply -f astra_control_center_min.yaml -n [netapp-acc or custom
   namespace]
   ```

   샘플 반응:

   ```
   astracontrolcenter.astra.netapp.io/astra created
   ```

시스템 상태를 확인합니다

> OpenShift를 사용하려는 경우 검증 단계에 유사한 OC 명령을 사용할 수 있습니다.

1. 모든 시스템 구성 요소가 성공적으로 설치되었는지 확인합니다.

```
kubectl get pods -n [netapp-acc or custom namespace]
```

   각 POD의 상태는 입니다 `Running`. 시스템 포드를 구축하는 데 몇 분 정도 걸릴 수 있습니다.

샘플 응답

```
NAME                                        READY   STATUS    RESTARTS
AGE
acc-helm-repo-6b44d68d94-d8m55              1/1     Running   0
13m
activity-78f99ddf8-hltct                    1/1     Running   0
10m
api-token-authentication-457nl              1/1     Running   0
9m28s
api-token-authentication-dgwsz              1/1     Running   0
9m28s
api-token-authentication-hmqqc              1/1     Running   0
9m28s
asup-75fd554dc6-m6qzh                       1/1     Running   0
9m38s
authentication-6779b4c85d-92gds             1/1     Running   0
8m11s
bucketservice-7cc767f8f8-lqwr8              1/1     Running   0
9m31s
certificates-549fd5d6cb-5kmd6               1/1     Running   0
9m56s
certificates-549fd5d6cb-bkjh9               1/1     Running   0
9m56s
cloud-extension-7bcb7948b-hn8h2             1/1     Running   0
10m
cloud-insights-service-56ccf86647-fgg69     1/1     Running   0
9m46s
composite-compute-677685b9bb-7vgsf          1/1     Running   0
10m
composite-volume-657d6c5585-dnq79           1/1     Running   0
9m49s
credentials-755fd867c8-vrlmt                1/1     Running   0
11m
entitlement-86495cdf5b-nwhh2                1/1     Running   2
10m
features-5684fb8b56-8d6s8                   1/1     Running   0
10m
fluent-bit-ds-rhx7v                         1/1     Running   0
7m48s
fluent-bit-ds-rjms4                         1/1     Running   0
7m48s
fluent-bit-ds-zf5ph                         1/1     Running   0
7m48s
graphql-server-66d895f544-w6hjd             1/1     Running   0
```

```
3m29s
identity-744df448d5-rlcmm                1/1     Running   0
10m
influxdb2-0                              1/1     Running   0
13m
keycloak-operator-75c965cc54-z7csw       1/1     Running   0
8m16s
krakend-798d6df96f-9z2sk                 1/1     Running   0
3m26s
license-5fb7d75765-f8mjg                 1/1     Running   0
9m50s
login-ui-7d5b7df85d-l2s7s                1/1     Running   0
3m20s
loki-0                                   1/1     Running   0
13m
metrics-facade-599b9d7fcc-gtmgl          1/1     Running   0
9m40s
monitoring-operator-67cc74f844-cdplp     2/2     Running   0
8m11s
nats-0                                   1/1     Running   0
13m
nats-1                                   1/1     Running   0
13m
nats-2                                   1/1     Running   0
12m
nautilus-769f5b74cd-k5jxm                1/1     Running   0
9m42s
nautilus-769f5b74cd-kd9gd                1/1     Running   0
8m59s
openapi-84f6ccd8ff-76kvp                 1/1     Running   0
9m34s
packages-6f59fc67dc-4g2f5                1/1     Running   0
9m52s
polaris-consul-consul-server-0           1/1     Running   0
13m
polaris-consul-consul-server-1           1/1     Running   0
13m
polaris-consul-consul-server-2           1/1     Running   0
13m
polaris-keycloak-0                       1/1     Running   0
8m7s
polaris-keycloak-1                       1/1     Running   0
5m49s
polaris-keycloak-2                       1/1     Running   0
5m15s
polaris-keycloak-db-0                    1/1     Running   0
```

```
8m6s
polaris-keycloak-db-1                     1/1     Running   0
5m49s
polaris-keycloak-db-2                     1/1     Running   0
4m57s
polaris-mongodb-0                         2/2     Running   0
13m
polaris-mongodb-1                         2/2     Running   0
12m
polaris-mongodb-2                         2/2     Running   0
12m
polaris-ui-565f56bf7b-zwr8b               1/1     Running   0
3m19s
polaris-vault-0                           1/1     Running   0
13m
polaris-vault-1                           1/1     Running   0
13m
polaris-vault-2                           1/1     Running   0
13m
public-metrics-6d86d66444-2wbzl           1/1     Running   0
9m30s
storage-backend-metrics-77c5d98dcd-dbhg5 1/1     Running   0
9m44s
storage-provider-78c885f57c-6zcv4         1/1     Running   0
9m36s
telegraf-ds-2l2m9                         1/1     Running   0
7m48s
telegraf-ds-qfzgh                         1/1     Running   0
7m48s
telegraf-ds-shrms                         1/1     Running   0
7m48s
telegraf-rs-bjpkt                         1/1     Running   0
7m48s
telemetry-service-6684696c64-qzfdf        1/1     Running   0
10m
tenancy-6596b6c54d-vmpsm                  1/1     Running   0
10m
traefik-7489dc59f9-6mnst                  1/1     Running   0
3m19s
traefik-7489dc59f9-xrkgg                  1/1     Running   0
3m4s
trident-svc-6c8dc458f5-jswcl              1/1     Running   0
10m
vault-controller-6b954f9b76-gz9nm         1/1     Running   0
11m
```

2. (선택 사항) 설치가 완료되었는지 확인하기 위해 을(를) 볼 수 있습니다 `acc-operator` 다음 명령을 사용하여 기록합니다.

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

> `accHost` 클러스터 등록은 마지막 작업 중 하나이며, 클러스터 등록에 실패하면 배포에 실패하지 않습니다. 로그에 클러스터 등록 실패가 표시되는 경우 클러스터 추가 워크플로우를 통해 등록을 다시 시도할 수 있습니다 "를 클릭합니다" API를 사용합니다.

3. 모든 Pod가 실행되면 설치가 성공적으로 완료되었는지 확인합니다 (`READY` 있습니다 `True`)를 입력하고 Astra Control Center에 로그인할 때 사용할 일회용 암호를 받습니다.

```
kubectl get AstraControlCenter -n netapp-acc
```

응답:

```
NAME    UUID                                        VERSION     ADDRESS
READY
astra   ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  22.08.1-26
10.111.111.111  True
```

> UUID 값을 복사합니다. 암호는 입니다 `ACC-` UUID 값 뒤에 옵니다 (`ACC-[UUID]` 또는, 이 예에서는 `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`)를 클릭합니다.

부하 분산을 위한 수신 설정

클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 컨트롤러를 설정할 수 있습니다.

이 절차에서는 수신 컨트롤러를 설정하는 방법에 대해 설명합니다 (`ingressType:Generic`)를 클릭합니다. 이것은 Astra Control Center의 기본 동작입니다. Astra Control Center를 배포한 후 URL을 사용하여 Astra Control Center를 노출하도록 수신 컨트롤러를 구성해야 합니다.

> 수신 컨트롤러를 설정하지 않으려면 을 설정할 수 있습니다 `ingressType:AccTraefik)`. Astra Control Center는 "loadbalancer" 유형의 서비스를 사용합니다. (`svc/traefik` Astra Control Center 네임스페이스에서), 액세스 가능한 외부 IP 주소를 할당해야 합니다. 로드 밸런서가 사용자 환경에서 허용되고 아직 로드 밸런서가 구성되어 있지 않은 경우 MetalLB 또는 다른 외부 서비스 로드 밸런서를 사용하여 외부 IP 주소를 서비스에 할당할 수 있습니다. 내부 DNS 서버 구성에서 Astra Control Center에 대해 선택한 DNS 이름을 부하 분산 IP 주소로 지정해야 합니다. "로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을 참조하십시오 "요구 사항".

단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다.

- 이스티오 침투

- Nginx 수신 컨트롤러
- OpenShift 수신 컨트롤러

필요한 것

- 필수 요소입니다 "수신 컨트롤러" 이미 배포되어 있어야 합니다.
- 를 클릭합니다 "수신 클래스" 수신 컨트롤러에 해당하는 컨트롤러가 이미 생성되어야 합니다.
- V1.19 및 v1.22 등의 Kubernetes 버전을 사용하고 있습니다.

**Istio** 침투에 대한 단계

1. Istio Ingress를 구성합니다.

   이 절차에서는 "기본" 구성 프로파일을 사용하여 Istio를 구축한다고 가정합니다.

2. 수신 게이트웨이에 대해 원하는 인증서 및 개인 키 파일을 수집하거나 생성합니다.

   CA 서명 또는 자체 서명 인증서를 사용할 수 있습니다. 공통 이름은 Astra 주소(FQDN)여야 합니다.

   명령 예:

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048
   -keyout tls.key -out tls.crt
   ```

3. 암호를 만듭니다 `tls secret name` 유형 `kubernetes.io/tls` 에서 TLS 개인 키 및 인증서의 경우 `istio-system namespace` TLS 비밀에 설명되어 있습니다.

   명령 예:

   ```
   kubectl create secret tls [tls secret name]
   --key="tls.key"
   --cert="tls.crt" -n istio-system
   ```

   비밀의 이름은 과 일치해야 합니다 `spec.tls.secretName` 에 제공됩니다 `istio-ingress.yaml` 파일.

4. 수신 리소스를 에 배포합니다 `netapp-acc` v1beta1(또는 1.22 미만의 Kubernetes 버전에서 사용되지 않음) 또는 v1 리소스 유형을 사용하는(또는 사용자 지정 이름) 네임스페이스입니다.

   출력:

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          serviceName: traefik
          servicePort: 80
```

v1 새 스키마의 경우 다음 샘플을 따르십시오.

```
kubectl apply -f istio-Ingress.yaml
```

출력:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. Astra Control Center를 평소처럼 배포합니다.
6. 수신 상태를 점검하십시오.

```
kubectl get ingress -n netapp-acc
```

응답:

```
NAME    CLASS HOSTS              ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

**Nginx** 수신 컨트롤러 단계

1. 형식의 암호를 만듭니다[`kubernetes.io/tls`] 에서 TLS 개인 키 및 인증서의 경우 `netapp-acc` 에 설명된

대로 (또는 사용자 지정 이름) 네임스페이스를 사용합니다 "TLS 비밀".

2. 수신 리소스를 에 배포합니다 `netapp-acc` 또는 사용자 지정 이름 네임스페이스 중 하나를 사용합니다 `v1beta1` (Kubernetes 버전 1.22 이하) 또는 에서는 사용되지 않습니다 `v1` 사용되지 않는 스키마나 새 스키마의 리소스 유형:

   a. 을(를) 위한 `v1beta1` 더 이상 사용되지 않는 스키마는 다음 샘플을 따르십시오.

```
apiVersion: extensions/v1beta1
Kind: IngressClass
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

   b. 의 경우 `v1` 새 스키마에 따라 다음 샘플을 수행합니다.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

**OpenShift Ingress** 컨트롤러를 위한 단계

1. 인증서를 구입하고 OpenShift 라우트에서 사용할 수 있도록 준비된 키, 인증서 및 CA 파일을 가져옵니다.
2. OpenShift 경로를 생성합니다.

```
oc create route edge --service=traefik
--port=web -n [netapp-acc or custom namespace]
--insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

**Astra Control Center UI**에 로그인합니다

Astra Control Center를 설치한 후 기본 관리자의 암호를 변경하고 Astra Control Center UI 대시보드에 로그인합니다.

단계

1. 브라우저에서 에 사용한 FQDN을 입력합니다 `astraAddress` 에 있습니다 `astra_control_center_min.yaml` CR Astra Control Center를 설치했습니다.
2. 메시지가 표시되면 자체 서명된 인증서를 수락합니다.

> 로그인 후 사용자 지정 인증서를 만들 수 있습니다.

3. Astra Control Center 로그인 페이지에서 에 사용한 값을 입력합니다 `email` 인치 `astra_control_center_min.yaml` CR Astra Control Center를 설치했습니다1회 암호 뒤에 옵니다 (`ACC-[UUID]`)를 클릭합니다.

> ⓘ 잘못된 암호를 세 번 입력하면 15분 동안 관리자 계정이 잠깁니다.

4. Login * 을 선택합니다.
5. 메시지가 나타나면 암호를 변경합니다.

> ⓘ 처음 로그인하는 데 암호를 잊은 경우 다른 관리 사용자 계정이 아직 생성되지 않은 경우 NetApp 지원에 암호 복구 지원을 문의하십시오.

6. (선택 사항) 기존의 자체 서명된 TLS 인증서를 제거하고 로 바꿉니다 "인증 기관(CA)에서 서명한 사용자 지정 TLS 인증서".

설치 문제를 해결합니다

에 서비스가 있는 경우 `Error` 상태, 로그를 검사할 수 있습니다. 400 ~ 500 범위의 API 응답 코드를 찾습니다. 이는 고장이 발생한 장소를 나타냅니다.

단계

1. Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

```
kubectl logs --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name) -c manager
```

다음 단계

를 수행하여 배포를 완료합니다 "설정 작업".

=
:allow-uri-read:

**POD** 보안 정책 제한 사항 이해

Astra Control Center는 POD 보안 정책(PSP)을 통해 권한 제한을 지원합니다. POD 보안 정책을 사용하면 컨테이너를 실행할 수 있는 사용자 또는 그룹과 해당 컨테이너에 사용할 수 있는 권한을 제한할 수 있습니다.

RKE2와 같은 일부 Kubernetes 배포에는 기본 POD 보안 정책이 너무 제한적이며 Astra Control Center 설치 시 문제가 발생합니다.

여기에 포함된 정보와 예제를 사용하여 Astra Control Center가 생성하는 POD 보안 정책을 이해하고, Astra Control Center 기능을 방해하지 않으면서 필요한 보호 기능을 제공하는 POD 보안 정책을 구성할 수 있습니다.

#### **Astra Control Center**에서 설치한 **PSP**

Astra Control Center는 설치 중에 몇 가지 POD 보안 정책을 생성합니다. 이 중 일부는 영구적이며 일부 작업은 특정 작업 중에 생성되며 작업이 완료되면 제거됩니다.

설치 중에 **PSP**가 생성되었습니다

Astra Control Center를 설치하는 동안 Astra Control Center 운영자는 사용자 지정 POD 보안 정책, 역할 개체 및 RoleBinding 개체를 설치하여 Astra Control Center 네임스페이스에 Astra Control Center 서비스를 배포할 수 있도록 합니다.

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

NAME                            PRIV    CAPS          SELINUX    RUNASUSER
FSGROUP     SUPGROUP    READONLYROOTFS   VOLUMES
avp-psp                         false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *
netapp-astra-deployment-psp     false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *

kubectl get role

NAME                                     CREATED AT
netapp-astra-deployment-role             2022-06-27T19:34:58Z

kubectl get rolebinding

NAME                                     ROLE
AGE
netapp-astra-deployment-rb               Role/netapp-astra-deployment-role
32m
```

백업 작업 중에 **PSP**가 생성되었습니다

백업 작업 중에 Astra Control Center는 동적 POD 보안 정책, ClusterRole 개체 및 RoleBinding 개체를 만듭니다. 이러한 백업 프로세스는 별도의 네임스페이스에서 수행됩니다.

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

NAME                            PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-astra-backup             false   DAC_READ_SEARCH
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false            *

kubectl get role

NAME                   CREATED AT
netapp-astra-backup    2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                   ROLE                        AGE
netapp-astra-backup    Role/netapp-astra-backup    62s
```

클러스터 관리 중에 생성된 **PSP**입니다

클러스터를 관리할 때 Astra Control Center는 NetApp 모니터링 연산자를 관리형 클러스터에 설치합니다. 이 연산자는 POD 보안 정책, ClusterRole 개체 및 RoleBinding 개체를 만들어 Astra Control Center 네임스페이스에 원격 측정 서비스를 배포합니다.

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

NAME                            PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-monitoring-psp-nkmo      true    AUDIT_WRITE,NET_ADMIN,NET_RAW
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false            *

kubectl get role

NAME                                          CREATED AT
netapp-monitoring-role-privileged             2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                                                  ROLE
AGE
netapp-monitoring-role-binding-privileged             Role/netapp-
monitoring-role-privileged              2m5s
```

네임스페이스 간 네트워크 통신을 활성화합니다

일부 환경에서는 NetworkPolicy 구문을 사용하여 네임스페이스 간 트래픽을 제한합니다. Astra Control Center 운영자, Astra Control Center 및 VMware vSphere용 Astra Plugin은 모두 서로 다른 네임스페이스에 있습니다. 서로 다른 네임스페이스에 있는 서비스는 서로 통신할 수 있어야 합니다. 이 통신을 활성화하려면 다음 단계를 수행하십시오.

단계

1. Astra Control Center 네임스페이스에 있는 NetworkPolicy 리소스를 삭제합니다.

```
kubectl get networkpolicy -n netapp-acc
```

2. 앞의 명령으로 반환된 각 NetworkPolicy 개체에 대해 다음 명령을 사용하여 개체를 삭제합니다. object_name>을(를) 반환된 개체의 이름으로 바꿉니다.

```
kubectl delete networkpolicy <OBJECT_NAME> -n netapp-acc
```

3. Astra Plugin for VMware vSphere 서비스가 Astra Control Center 서비스에 요청을 할 수 있도록 acc-AVP-network-policy 객체를 구성하려면 다음 리소스 파일을 적용하십시오. 괄호 <>의 정보를 사용자 환경의 정보로 바꿉니다.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-avp-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <PLUGIN_NAMESPACE_NAME> #
REPLACE THIS WITH THE ASTRA PLUGIN FOR VMWARE VSPHERE NAMESPACE NAME
```

4. 다음 리소스 파일을 적용하여 Astra Control Center 운영자가 Astra Control Center 서비스와 통신할 수 있도록 acc-operator-network-policy 객체를 구성합니다. 괄호 <>의 정보를 사용자 환경의 정보로 바꿉니다.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-operator-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <NETAPP-ACC-OPERATOR> #
REPLACE THIS WITH THE OPERATOR NAMESPACE NAME
```

리소스 제한을 제거합니다

일부 환경에서는 ResourceQuotas 및 LimitRanges 개체를 사용하여 네임스페이스의 리소스가 클러스터에서 사용 가능한 모든 CPU 및 메모리를 사용하지 못하도록 합니다. Astra Control Center는 최대 제한을 설정하지 않으므로 해당 리소스를 준수하지 않습니다. Astra Control Center를 설치할 네임스페이스에서 제거해야 합니다.

다음 단계를 사용하여 할당량 및 제한을 검색하고 제거할 수 있습니다. 이 예제에서 명령 출력은 명령 직후에 표시됩니다.

단계

1. NetApp-acc 네임스페이스에서 리소스 할당량 확인:

```
kubectl get quota -n netapp-acc
```

응답:

```
NAME          AGE   REQUEST                                          LIMIT
pods-high     16s   requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low      15s   requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium   16s   requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. 이름으로 모든 리소스 할당량 삭제:

```
kubectl delete resourcequota  pods-high -n netapp-acc
```

```
kubectl delete resourcequota  pods-low -n netapp-acc
```

```
kubectl delete resourcequota  pods-medium -n netapp-acc
```

3. NetApp-acc 네임스페이스의 제한 범위를 가져옵니다.

```
kubectl get limits -n netapp-acc
```

응답:

```
NAME               CREATED AT
cpu-limit-range    2022-06-27T19:01:23Z
```

4. 이름별로 제한 범위를 삭제합니다.

```
kubectl delete limitrange cpu-limit-range -n netapp-acc
```

=
:allow-uri-read:

## OpenShift OperatorHub를 사용하여 Astra Control Center를 설치합니다

Red Hat OpenShift를 사용하는 경우 Red Hat 공인 운영자를 사용하여 Astra Control Center를 설치할 수 있습니다. 이 절차를 사용하여 에서 Astra Control Center를 설치합니다 "Red Hat 에코시스템 카탈로그" 또는 Red Hat OpenShift Container Platform 사용.

이 절차를 완료한 후에는 설치 절차로 돌아가 를 완료해야 합니다 "나머지 단계" 설치 성공 여부를 확인하고 로그온합니다.

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- OpenShift 클러스터에서 모든 클러스터 운영자가 양호한 상태인지 확인합니다 (`available` 있습니다 `true`):

```
oc get clusteroperators
```

- OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다 (`available` 있습니다 `true`):

```
oc get apiservices
```

- 데이터 센터에서 Astra Control Center에 대한 FQDN 주소를 생성합니다.
- 필요한 사용 권한을 얻고 Red Hat OpenShift Container Platform에 액세스하여 설명된 설치 단계를 수행합니다.
- 클러스터에 인증서 관리자가 이미 있는 경우 일부를 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하지 않습니다.

단계

- Astra Control Center 번들을 다운로드하고 포장을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 운영자 설치 페이지를 찾으십시오
- 운전자를 설치합니다
- Astra Control Center를 설치합니다

### **Astra Control Center** 번들을 다운로드하고 포장을 풉니다

1. Astra Control Center 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "NetApp Support 사이트".
2. 에서 Astra Control Center 인증서 및 키의 지퍼를 다운로드합니다 "NetApp Support 사이트".
3. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

```
openssl dgst -sha256 -verify AstraControlCenter-public.pub -signature
astra-control-center-[version].tar.gz.sig astra-control-center-
[version].tar.gz
```

4. 이미지 추출:

```
tar -vxzf astra-control-center-[version].tar.gz
```

### **NetApp Astra kubtl** 플러그인을 설치합니다

NetApp 아스트라 `kubectl` 명령줄 플러그인을 사용하면 Astra Control Center 배포 및 업그레이드와 관련된 일반적인 작업을 수행할 때 시간을 절약할 수 있습니다.

필요한 것

NetApp은 다양한 CPU 아키텍처 및 운영 체제용 플러그인의 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다. Linux 및 Mac 운영 체제에서는 를 사용할 수 있습니다 `uname -a` 명령을 사용하여 이 정보를 수집합니다.

단계

1. 사용 가능한 NetApp Astra를 나열하십시오 `kubectl` 플러그인 바이너리를 만들고 운영 체제 및 CPU 아키텍처에 필요한 파일 이름을 적어 둡니다.

```
ls kubectl-astra/
```

2. 파일을 규격과 같은 위치에 복사합니다 `kubectl` 유틸리티. 이 예에서 는 입니다 `kubectl` 유틸리티는 에 있습니다 `/usr/local/bin` 디렉토리. 대치 `<binary-name>` 필요한 파일 이름:

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

이미지를 로컬 레지스트리에 추가합니다

1. 용기 엔진에 적합한 단계 시퀀스를 완료합니다.

**Docker** 를 참조하십시오

1. Astra 디렉토리로 이동합니다.

```
cd acc
```

2. Astra Control Center 이미지 디렉토리에 있는 패키지 이미지를 로컬 레지스트리로 푸시합니다. 명령을 실행하기 전에 다음 대체 작업을 수행합니다.
   - Bundle_file을 Astra Control 번들 파일 이름으로 바꿉니다(예: `acc.manifest.yaml`)를 클릭합니다.
   - my_registry를 Docker 리포지토리의 URL로 바꿉니다.
   - my_registry_user를 사용자 이름으로 바꿉니다.
   - my_registry_token을 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m BUNDLE_FILE -r MY_REGISTRY
-u MY_REGISTRY_USER -p MY_REGISTRY_TOKEN
```

팟맨

1. 레지스트리에 로그인합니다.

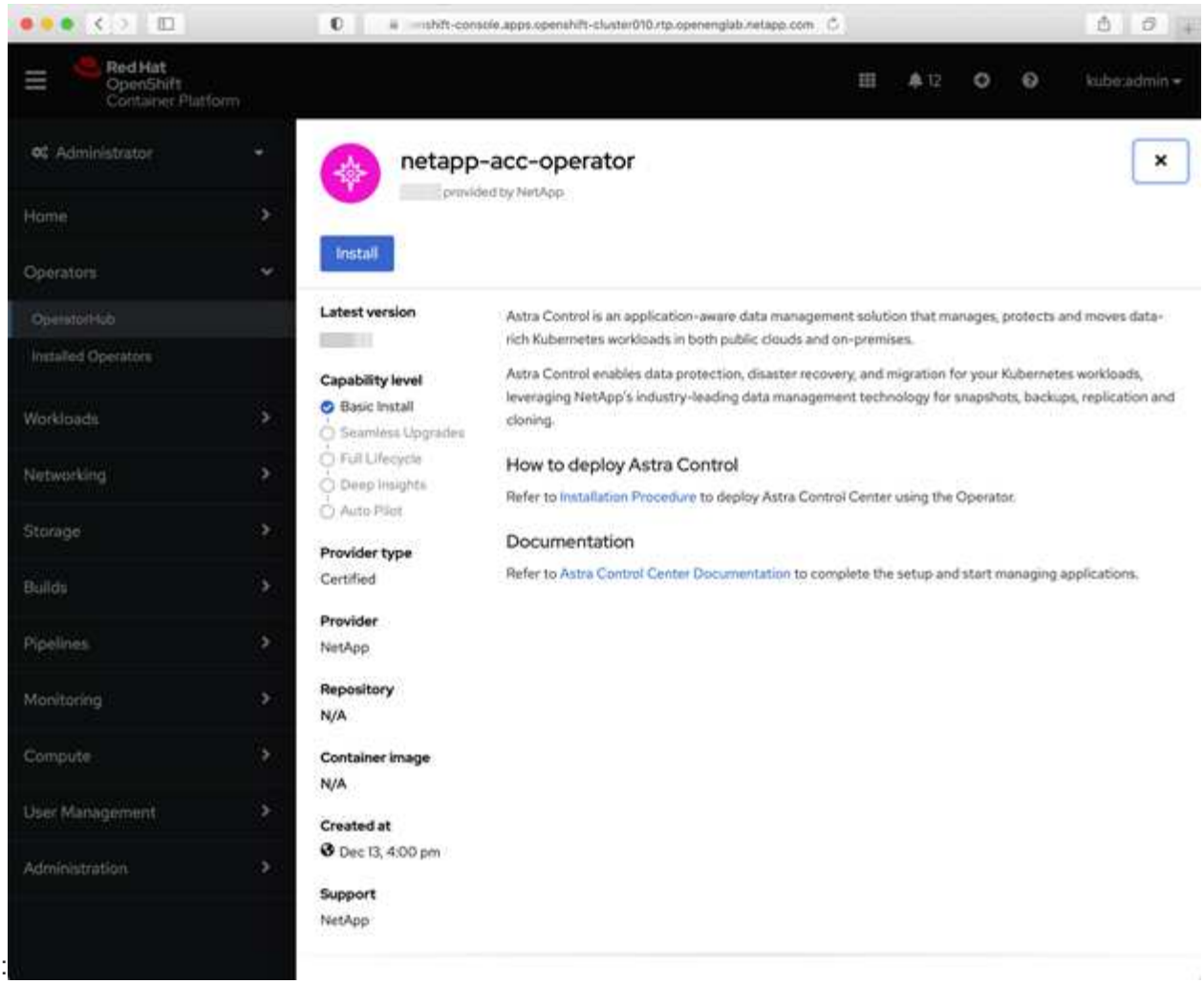
```
podman login [your_registry_path]
```

2. 설명에 명시된 대로 <your_registry> 대체를 만들어 다음 스크립트를 실행합니다.

```
# You need to be at the root of the tarball.
# You should see these files to confirm correct location:
#   acc.manifest.yaml
#   acc/

# Replace <YOUR_REGISTRY> with your own registry (e.g
registry.customer.com or registry.customer.com/testing, etc..)
export REGISTRY=<YOUR_REGISTRY>
export PACKAGENAME=acc
export PACKAGEVERSION=22.08.1-26
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
  # Load to local cache
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')

  # Remove path and keep imageName.
  astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')

  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/netapp/astra/${PACKAGENAME}
/${PACKAGEVERSION}/${astraImageNoPath}

  # Push to the local repo.
  podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

운영자 설치 페이지를 찾으십시오

1. 운영자 설치 페이지에 액세스하려면 다음 절차 중 하나를 완료하십시오.
   - Red Hat OpenShift 웹 콘솔




:

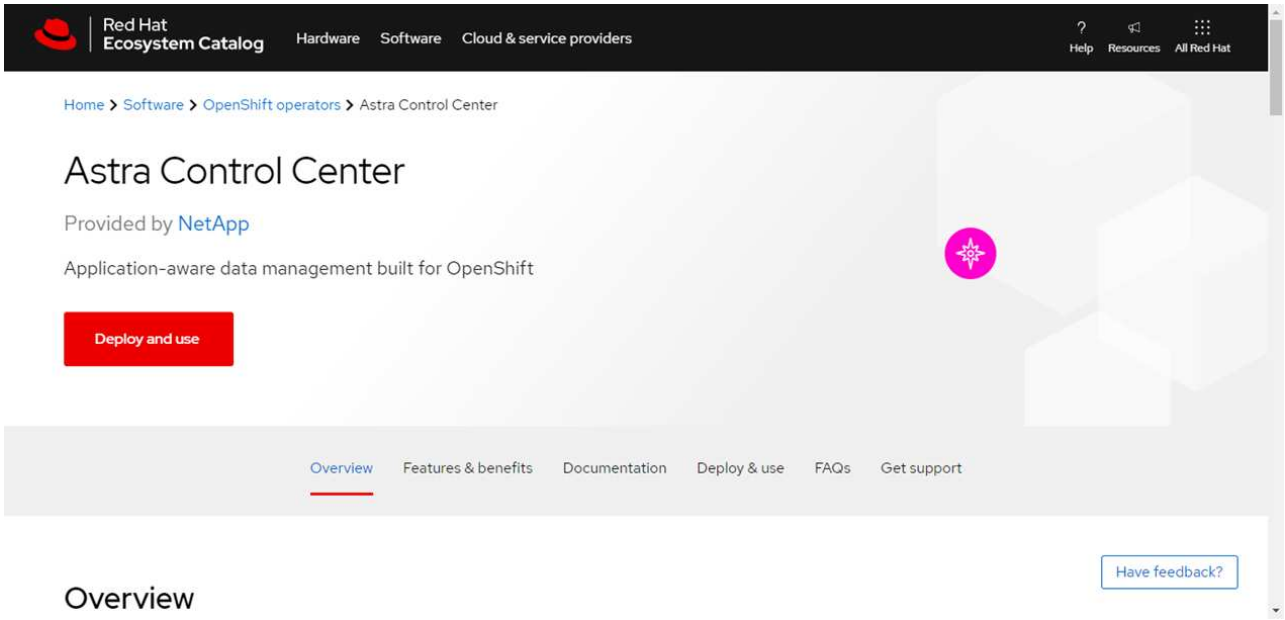
i. OpenShift Container Platform UI에 로그인합니다.

ii. 측면 메뉴에서 * Operators > OperatorHub * 를 선택합니다.

iii. NetApp Astra Control Center 운영자를 선택합니다.

iv. 설치 * 를 선택합니다.

- Red Hat 에코시스템 카탈로그
  :




i. NetApp Astra Control Center를 선택합니다 "운영자".

ii. 배포 및 사용 * 을 선택합니다.

운전자를 설치합니다

1. Install Operator * 페이지를 완료하고 운영자를 설치합니다.

> 운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

   a. 연산자 네임스페이스 또는 를 선택합니다 `netapp-acc-operator` 네임스페이스는 운영자 설치의 일부로 자동으로 생성됩니다.

   b. 수동 또는 자동 승인 전략을 선택합니다.

> 수동 승인이 권장됩니다. 클러스터당 하나의 운영자 인스턴스만 실행 중이어야 합니다.

   c. 설치 * 를 선택합니다.

> 수동 승인 전략을 선택한 경우 이 운영자에 대한 수동 설치 계획을 승인하라는 메시지가 표시됩니다.

2. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

**Astra Control Center**를 설치합니다

1. Astra Control Center 운용자의 세부 정보 보기 내 콘솔에서 를 선택합니다 `Create instance` 제공된 API 섹션에서

2. 를 완료합니다 `Create AstraControlCenter` 양식 필드:

   a. Astra Control Center 이름을 유지하거나 조정합니다.

   b. (선택 사항) 자동 지원을 활성화 또는 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.

   c. Astra Control Center 주소를 입력합니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

d. Astra Control Center 버전을 입력합니다(예: 21.12.60).

e. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.

f. 기본 볼륨 재확보 정책을 유지합니다.

g. 이미지 레지스트리 * 에서 로컬 컨테이너 이미지 레지스트리 경로를 입력합니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

h. 인증이 필요한 레지스트리를 사용하는 경우 암호를 입력합니다.

i. 관리자의 이름을 입력합니다.

j. 리소스 확장을 구성합니다.

k. 기본 스토리지 클래스를 유지합니다.

l. CRD 처리 기본 설정을 정의합니다.

3. 를 선택합니다 `Create`.

다음 단계

Astra Control Center가 성공적으로 설치되었는지 확인하고 를 완료합니다 "나머지 단계" 를 눌러 로그인합니다. 또한 를 수행하여 배포를 완료합니다 "설정 작업".

## Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치합니다

Astra Control Center를 사용하면 자체 관리되는 Kubernetes 클러스터 및 Cloud Volumes ONTAP 인스턴스가 있는 하이브리드 클라우드 환경에서 앱을 관리할 수 있습니다. 온프레미스 Kubernetes 클러스터 또는 클라우드 환경의 자가 관리 Kubernetes 클러스터 중 하나에 Astra Control Center를 구축할 수 있습니다.

이러한 구축 중 하나를 통해 Cloud Volumes ONTAP를 스토리지 백엔드로 사용하여 애플리케이션 데이터 관리 작업을 수행할 수 있습니다. S3 버킷을 백업 타겟으로 구성할 수도 있습니다.

AWS(Amazon Web Services), GCP(Google Cloud Platform) 및 Microsoft Azure에 Cloud Volumes ONTAP 스토리지 백엔드를 사용하여 Astra Control Center를 설치하려면 클라우드 환경에 따라 다음 단계를 수행하십시오.

- Amazon Web Services에 Astra Control Center를 구축합니다
- Google Cloud Platform에 Astra Control Center를 구축합니다
- Microsoft Azure에 Astra Control Center를 구축합니다

OCP(OpenShift Container Platform)와 같이 자체 관리되는 Kubernetes 클러스터를 사용하여 배포판에서 앱을 관리할 수 있습니다. 자가 관리 OCP 클러스터만 Astra Control Center 구축을 위해 검증되었습니다.

### Amazon Web Services에 Astra Control Center를 구축합니다

AWS(Amazon Web Services) 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

#### AWS에 필요한 것

AWS에 Astra Control Center를 구축하기 전에 다음 항목이 필요합니다.

- Astra Control Center 라이센스. 을 참조하십시오 "Astra Control Center 라이센스 요구 사항".

- "Astra Control Center 요구 사항을 충족합니다".
- NetApp Cloud Central 계정
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 AWS 자격 증명, 액세스 ID 및 비밀 키
- AWS 계정 ECR(Elastic Container Registry) 액세스 및 로그인
- Astra Control UI에 액세스하려면 AWS 호스팅 영역 및 Route 53 항목이 필요합니다

**AWS**의 운영 환경 요구사항

Astra Control Center에는 AWS를 위한 다음과 같은 운영 환경이 필요합니다.

- Red Hat OpenShift Container Platform 4.8

> Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다.

Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

| 구성 요소 | 요구 사항 |
|---|---|
| 백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다 | 최소 300GB가 사용 가능합니다 |
| 작업자 노드**(AWS EC2** 요구사항**)** | 총 3개 이상의 작업자 노드, vCPU 코어 4개, 12GB RAM |
| 로드 밸런서 | 수신 트래픽을 운영 환경 클러스터의 서비스로 전송할 수 있도록 서비스 유형 "로드 밸런서"를 사용할 수 있습니다 |
| **FQDN** | Astra Control Center의 FQDN을 부하 분산 IP 주소로 가리키는 방법 |
| **Astra Trident(NetApp Cloud Manager**에서 **Kubernetes** 클러스터 검색의 일부로 설치됨**)** | Astra Trident 21.04 이상 설치 및 구성, NetApp ONTAP 버전 9.5 이상 버전을 스토리지 백엔드로 사용합니다 |
| 이미지 레지스트리 | Astra Control Center 빌드 이미지를 푸시할 수 있는 AWS Elastic Container Registry와 같은 기존 개인 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다.<br><br>Astra Control Center에서 호스팅되는 클러스터와 관리 클러스터는 Resetic 기반 이미지를 사용하여 앱을 백업 및 복원할 수 있도록 동일한 이미지 레지스트리에 액세스할 수 있어야 합니다. |

| 구성 요소 | 요구 사항 |
| --- | --- |
| **Astra Trident/ONTAP** 구성 | Astra Control Center에서는 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Kubernetes 클러스터를 NetApp Cloud Manager로 가져올 때 생성되는 다음과 같은 ONTAP Kubernetes 스토리지 클래스를 지원합니다. Astra Trident에서 제공합니다.<br>• `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-san csi.trident.netapp.io` |

이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

AWS 레지스트리 토큰은 12시간 후에 만료되며, 그 후에는 Docker 이미지 레지스트리 암호를 갱신해야 합니다.

#### AWS 구축 개요

Cloud Volumes ONTAP를 스토리지 백엔드로 사용하여 Astra Control Center for AWS를 설치하는 프로세스를 간략하게 소개합니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. IAM 권한이 충분한지 확인하십시오.
2. AWS에 RedHat OpenShift 클러스터를 설치합니다.
3. AWS 구성.
4. NetApp Cloud Manager 구성.
5. Astra Control Center를 설치합니다.

#### IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp Cloud Manager Connector를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "초기 AWS 자격 증명".

#### AWS에 RedHat OpenShift 클러스터를 설치합니다

AWS에 RedHat OpenShift Container Platform 클러스터를 설치합니다.

설치 지침은 를 참조하십시오 "OpenShift Container Platform에서 AWS에 클러스터 설치".

**AWS** 구성

그런 다음 AWS를 구성하여 가상 네트워크를 생성하고, EC2 컴퓨팅 인스턴스를 설정하고, AWS S3 버킷을 생성하고, ECR(Elastic Container Register)을 생성하여 Astra Control Center 이미지를 호스팅하고, 이 레지스트리로 이미지를 푸시합니다.

AWS 설명서에 따라 다음 단계를 완료하십시오. 을 참조하십시오 "AWS 설치 설명서".

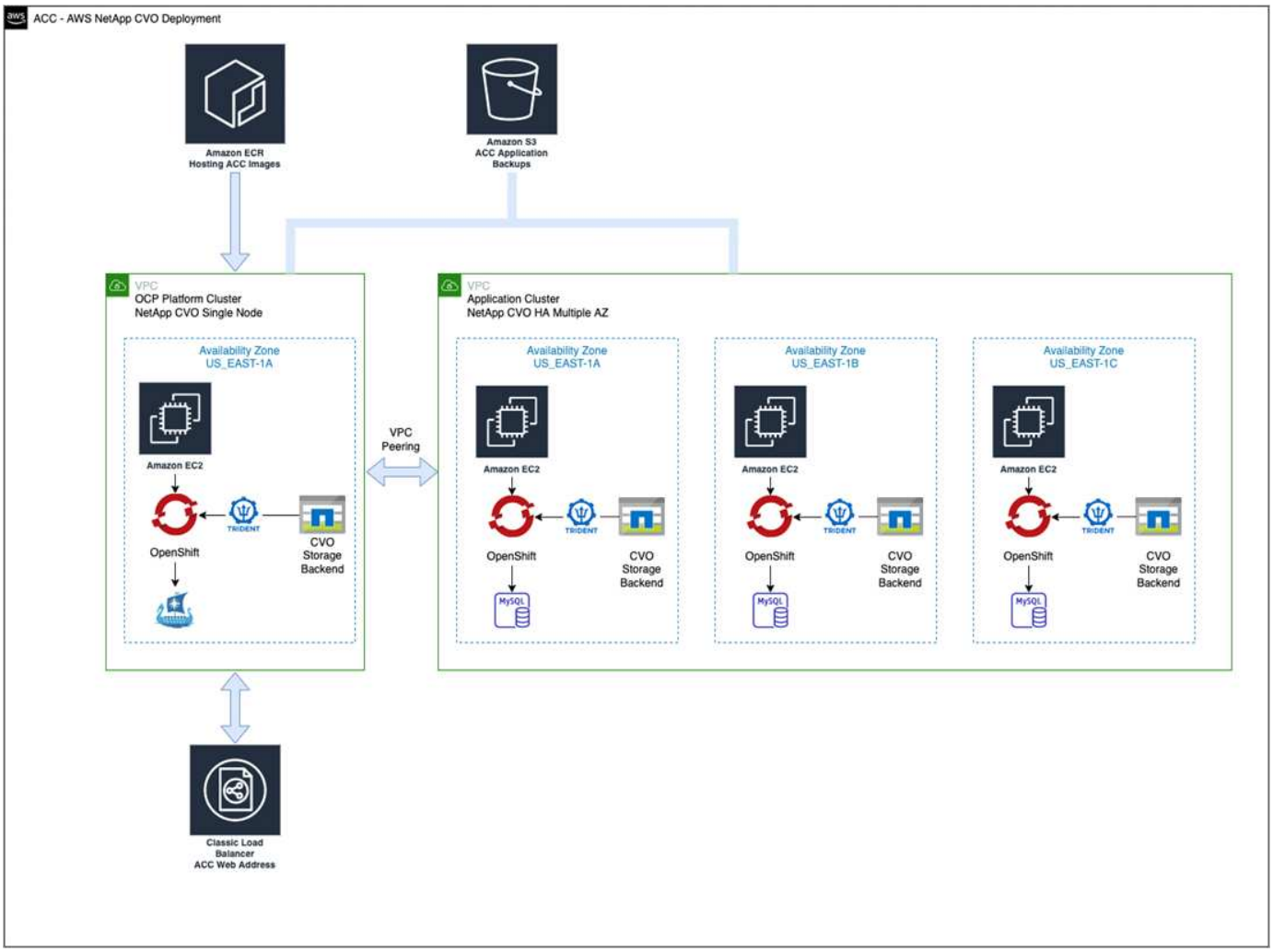
1. AWS 가상 네트워크를 생성합니다.
2. EC2 컴퓨팅 인스턴스를 검토합니다. 이는 AWS의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 AWS의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 AWS S3 버킷을 하나 이상 생성합니다.
5. AWS ECR(Elastic Container Registry)을 생성하여 모든 ACC 이미지를 호스팅합니다.

> ECR을 생성하지 않으면 Astra Control Center는 AWS 백엔드가 있는 Cloud Volumes ONTAP가 포함된 클러스터에서 모니터링 데이터에 액세스할 수 없습니다. 이 문제는 Astra Control Center를 사용하여 검색 및 관리하려는 클러스터에 AWS ECR 액세스 권한이 없을 때 발생합니다.

6. ACC 이미지를 정의된 레지스트리로 푸시합니다.

> AWS ECR(Elastic Container Registry) 토큰이 12시간 후에 만료되어 클러스터 간 클론 작업이 실패합니다. 이 문제는 AWS용으로 구성된 Cloud Volumes ONTAP에서 스토리지 백엔드를 관리할 때 발생합니다. 이 문제를 해결하려면 ECR을 다시 인증하고 클론 작업이 성공적으로 재개되도록 새로운 암호를 생성하십시오.

다음은 AWS 구축의 예입니다.




**NetApp Cloud Manager** 구성

Cloud Manager를 사용하여 작업 공간을 생성하고, AWS에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

Cloud Manager 설명서에 따라 다음 단계를 완료하십시오. 다음을 참조하십시오.

- "AWS에서 Cloud Volumes ONTAP 시작하기".
- "Cloud Manager를 사용하여 AWS에서 커넥터를 생성합니다"

단계

1. Cloud Manager에 자격 증명을 추가합니다.
2. 작업 영역을 만듭니다.
3. AWS용 커넥터를 추가합니다. AWS를 공급자로 선택합니다.
4. 클라우드 환경을 위한 작업 환경을 구축합니다.
    a. 위치: "AWS(Amazon Web Services)"
    b. 유형: "Cloud Volumes ONTAP HA"
5. OpenShift 클러스터를 가져옵니다. 클러스터가 방금 생성한 작업 환경에 연결됩니다.

a. NetApp 클러스터 세부 정보를 보려면 * K8s * > * 클러스터 목록 * > * 클러스터 세부 정보 * 를 선택합니다.

b. 오른쪽 위 모서리에서 Trident 버전을 확인합니다.

c. NetApp을 공급자 로 보여주는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 참조하십시오.

   그러면 Red Hat OpenShift 클러스터가 가져와 기본 스토리지 클래스가 할당됩니다. 스토리지 클래스를 선택합니다. Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

6. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.

> Cloud Volumes ONTAP는 단일 노드 또는 고가용성으로 작동할 수 있습니다. HA가 활성화된 경우 AWS에서 실행 중인 HA 상태와 노드 구축 상태를 확인하십시오.

#### **Astra Control Center**를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

> AWS는 일반 S3 버킷 유형을 사용합니다.

### **Google Cloud Platform**에 **Astra Control Center**를 구축합니다

GCP(Google Cloud Platform) 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

#### **GCP**에 필요한 사항

GCP에 Astra Control Center를 구축하기 전에 다음 항목이 필요합니다.

- Astra Control Center 라이센스. 을 참조하십시오 "Astra Control Center 라이센스 요구 사항".
- "Astra Control Center 요구 사항을 충족합니다".
- NetApp Cloud Central 계정
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 4.10
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 GCP 서비스 계정

#### **GCP**의 운영 환경 요구 사항

> Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다.

Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

| 구성 요소 | 요구 사항 |
|---|---|
| 백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다 | 최소 300GB가 사용 가능합니다 |
| 작업자 노드**(GCP** 컴퓨팅 요구사항**)** | 총 3개 이상의 작업자 노드, vCPU 코어 4개, 12GB RAM |

| 구성 요소 | 요구 사항 |
| --- | --- |
| 로드 밸런서 | 수신 트래픽을 운영 환경 클러스터의 서비스로 전송할 수 있도록 서비스 유형 "로드 밸런서"를 사용할 수 있습니다 |
| **FQDN(GCP DNS** 영역**)** | Astra Control Center의 FQDN을 부하 분산 IP 주소로 가리키는 방법 |
| **Astra Trident(NetApp Cloud Manager**에서 **Kubernetes** 클러스터 검색의 일부로 설치됨**)** | Astra Trident 21.04 이상 설치 및 구성, NetApp ONTAP 버전 9.5 이상 버전을 스토리지 백엔드로 사용합니다 |
| 이미지 레지스트리 | Astra Control Center 빌드 이미지를 푸시할 수 있는 Google Container Registry와 같은 기존 개인 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다.<br>ⓘ 백업을 위해 Restic 이미지를 풀려면 익명 액세스를 설정해야 합니다. |
| **Astra Trident/ONTAP** 구성 | Astra Control Center에서는 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Kubernetes 클러스터를 NetApp Cloud Manager로 가져올 때 생성되는 다음과 같은 ONTAP Kubernetes 스토리지 클래스를 지원합니다. Astra Trident에서 제공합니다.<br>• `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-san csi.trident.netapp.io` |

ⓘ 이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

**GCP** 구축 개요

다음은 Astra Control Center를 스토리지 백엔드로 Cloud Volumes ONTAP를 사용하는 GCP의 자체 관리 OCP 클러스터에 설치하는 프로세스의 개요입니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. GCP에 RedHat OpenShift 클러스터를 설치합니다.
2. GCP 프로젝트 및 가상 프라이빗 클라우드를 생성합니다.
3. IAM 권한이 충분한지 확인하십시오.
4. GCP를 구성합니다.

5. NetApp Cloud Manager 구성.
6. Astra Control Center를 설치하고 구성합니다.

#### GCP에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 GCP에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 다음을 참조하십시오.

- "GCP에서 OpenShift 클러스터 설치"
- "GCP 서비스 계정 생성"

#### GCP 프로젝트 및 가상 프라이빗 클라우드를 생성합니다

하나 이상의 GCP 프로젝트 및 VPC(가상 프라이빗 클라우드)를 생성합니다.

> OpenShift는 자체 리소스 그룹을 생성할 수 있습니다. 또한 GCP VPC를 정의해야 합니다. OpenShift 설명서를 참조하십시오.

플랫폼 클러스터 리소스 그룹과 대상 애플리케이션 OpenShift 클러스터 리소스 그룹을 생성할 수 있습니다.

#### IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp Cloud Manager Connector를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "초기 GCP 자격 증명 및 권한".

#### GCP를 구성합니다

그런 다음 VPC를 생성하고, 컴퓨팅 인스턴스를 설정하고, Google Cloud Object Storage를 생성하고, Google Container Register를 생성하여 Astra Control Center 이미지를 호스팅하고, 이미지를 이 레지스트리로 푸시하도록 GCP를 구성합니다.

GCP 문서에 따라 다음 단계를 완료합니다. GCP에서 OpenShift 클러스터 설치를 참조하십시오.

1. CVO 백엔드가 있는 OCP 클러스터에 사용할 GCP에서 사용할 GCP 프로젝트 및 VPC를 GCP에서 생성합니다.
2. 컴퓨팅 인스턴스를 검토합니다. GCP의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 Astra 요구 사항을 충족하도록 GCP의 인스턴스 유형을 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 하나 이상의 GCP Cloud Storage Bucket을 생성합니다.
5. 버킷 액세스에 필요한 암호를 생성합니다.
6. 모든 Astra Control Center 이미지를 호스트하기 위해 Google Container Registry를 생성합니다.
7. 모든 Astra Control Center 이미지에 대해 Docker 푸시/풀용 Google Container Registry 액세스를 설정합니다.

   예: 다음 스크립트를 입력하여 ACC 이미지를 이 레지스트리로 푸시할 수 있습니다.

```
gcloud auth activate-service-account <service account email address>
--key-file=<GCP Service Account JSON file>
```

이 스크립트에는 Astra Control Center 매니페스트 파일과 Google Image 레지스트리 위치가 필요합니다.

예:

```
manifestfile=astra-control-center-<version>.manifest
GCP_CR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

8. DNS 존 설정

#### NetApp Cloud Manager 구성

Cloud Manager를 사용하여 작업 공간을 생성하고, GCP에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

Cloud Manager 설명서에 따라 다음 단계를 완료하십시오. 을 참조하십시오 "GCP에서 Cloud Volumes ONTAP 시작하기".

필요한 것

- 필요한 IAM 권한 및 역할을 사용하여 GCP 서비스 계정에 액세스합니다

단계

1. Cloud Manager에 자격 증명을 추가합니다. 을 참조하십시오 "GCP 계정 추가".
2. GCP용 커넥터를 추가합니다.
   a. 공급자로 "GCP"를 선택합니다.
   b. GCP 자격 증명을 입력합니다. 을 참조하십시오 "Cloud Manager에서 GCP에 커넥터 생성".
   c. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.
3. 클라우드 환경을 위한 작업 환경을 구축합니다.
   a. 위치:"GCP"
   b. 유형: "Cloud Volumes ONTAP HA"

4. OpenShift 클러스터를 가져옵니다. 클러스터가 방금 생성한 작업 환경에 연결됩니다.
   a. NetApp 클러스터 세부 정보를 보려면 * K8s * > * 클러스터 목록 * > * 클러스터 세부 정보 * 를 선택합니다.
   b. 오른쪽 위 모서리에서 Trident 버전을 확인합니다.
   c. "NetApp"을 프로비저닝자로 나타내는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 확인하십시오.

      그러면 Red Hat OpenShift 클러스터가 가져와 기본 스토리지 클래스가 할당됩니다. 스토리지 클래스를 선택합니다. Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.
5. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.

> Cloud Volumes ONTAP는 단일 노드 또는 고가용성(HA)으로 작동할 수 있습니다. HA가 사용되도록 설정된 경우 GCP에서 실행 중인 HA 상태 및 노드 배포 상태를 확인합니다.

#### **Astra Control Center**를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

> GCP는 일반 S3 버킷 유형을 사용합니다.

1. Docker Secret를 생성하여 Astra Control Center 설치를 위한 이미지를 가져옵니다.

```
kubectl create secret docker-registry <secret name>
--docker-server=<Registry location>
--docker-username=_json_key
--docker-password="$(cat <GCP Service Account JSON file>)"
--namespace=pcloud
```

### **Microsoft Azure**에 **Astra Control Center**를 구축합니다

Microsoft Azure 퍼블릭 클라우드에서 호스팅되는 자가 관리형 Kubernetes 클러스터에 Astra Control Center를 구축할 수 있습니다.

#### **Azure**에 필요한 기능

Azure에 Astra Control Center를 배포하기 전에 다음 항목이 필요합니다.

- Astra Control Center 라이센스. 을 참조하십시오 "Astra Control Center 라이센스 요구 사항".
- "Astra Control Center 요구 사항을 충족합니다".
- NetApp Cloud Central 계정
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 4.8
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 Azure 자격 증명

Astra Control Center를 호스팅하기 위해 선택한 운영 환경이 환경 공식 문서에 설명된 기본 리소스 요구 사항을 충족하는지 확인합니다.

Astra Control Center에는 환경의 리소스 요구 사항 외에 다음과 같은 리소스가 필요합니다.

을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

| 구성 요소 | 요구 사항 |
|---|---|
| 백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다 | 최소 300GB가 사용 가능합니다 |
| 작업자 노드**(Azure** 컴퓨팅 요구 사항**)** | 총 3개 이상의 작업자 노드, vCPU 코어 4개, 12GB RAM |
| 로드 밸런서 | 수신 트래픽을 운영 환경 클러스터의 서비스로 전송할 수 있도록 서비스 유형 "로드 밸런서"를 사용할 수 있습니다 |
| **FQDN(Azure DNS** 영역**)** | Astra Control Center의 FQDN을 부하 분산 IP 주소로 가리키는 방법 |
| **Astra Trident(NetApp Cloud Manager**에서 **Kubernetes** 클러스터 검색의 일부로 설치됨**)** | 설치 및 구성된 Astra Trident 21.04 이상 및 NetApp ONTAP 버전 9.5 이상이 스토리지 백엔드로 사용됩니다 |
| 이미지 레지스트리 | Astra Control Center 빌드 이미지를 푸시할 수 있는 Azure 컨테이너 레지스트리(ACR)와 같은 기존 개인 레지스트리가 있어야 합니다. 이미지를 업로드할 이미지 레지스트리의 URL을 제공해야 합니다.<br>ⓘ 백업을 위해 Restic 이미지를 풀려면 익명 액세스를 설정해야 합니다. |
| **Astra Trident/ONTAP** 구성 | Astra Control Center에서는 스토리지 클래스를 생성하고 기본 스토리지 클래스로 설정해야 합니다. Astra Control Center는 Kubernetes 클러스터를 NetApp Cloud Manager로 가져올 때 생성되는 다음과 같은 ONTAP Kubernetes 스토리지 클래스를 지원합니다. Astra Trident에서 제공합니다.<br>• `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io`<br>• `vsaworkingenvironment-<>-single-san csi.trident.netapp.io` |

ⓘ 이러한 요구 사항에서는 Astra Control Center가 운영 환경에서 실행되는 유일한 애플리케이션이라고 가정합니다. 환경에서 추가 애플리케이션이 실행 중인 경우 이러한 최소 요구 사항을 적절히 조정합니다.

#### Azure 구축 개요

다음은 Azure용 Astra Control Center를 설치하는 프로세스의 개요입니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. Azure에 RedHat OpenShift 클러스터를 설치합니다.
2. Azure 리소스 그룹을 생성합니다.
3. IAM 권한이 충분한지 확인하십시오.
4. Azure를 구성합니다.
5. NetApp Cloud Manager 구성.
6. Astra Control Center를 설치하고 구성합니다.

#### Azure에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 Azure에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 의 RedHat 설명서를 참조하십시오 "Azure에 OpenShift 클러스터 설치" 및 "Azure 계정을 설치하는 중입니다".

#### Azure 리소스 그룹을 생성합니다

Azure 리소스 그룹을 하나 이상 생성합니다.

> OpenShift는 자체 리소스 그룹을 생성할 수 있습니다. 또한 Azure 리소스 그룹을 정의해야 합니다. OpenShift 설명서를 참조하십시오.

플랫폼 클러스터 리소스 그룹과 대상 애플리케이션 OpenShift 클러스터 리소스 그룹을 생성할 수 있습니다.

#### IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp Cloud Manager Connector를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "Azure 자격 증명 및 권한".

#### Azure를 구성합니다

그런 다음 가상 네트워크를 만들고, 컴퓨팅 인스턴스를 설정하고, Azure Blob 컨테이너를 만들고, Astra Control Center 이미지를 호스팅하기 위해 ACR(Azure Container Register)을 만들고, 이 레지스트리로 이미지를 푸시하도록 Azure를 구성합니다.

Azure 설명서에 따라 다음 단계를 완료합니다. 을 참조하십시오 "Azure에 OpenShift 클러스터 설치".

1. Azure 가상 네트워크를 생성합니다.
2. 컴퓨팅 인스턴스를 검토합니다. Azure의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 Azure의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".

4. 백업을 저장할 Azure Blob 컨테이너를 하나 이상 생성합니다.
5. 저장소 계정을 생성합니다. Astra Control Center에서 버킷으로 사용할 컨테이너를 생성하려면 저장소 계정이 필요합니다.
6. 버킷 액세스에 필요한 암호를 생성합니다.
7. Azure Container Registry(ACR)를 생성하여 모든 Astra Control Center 이미지를 호스트합니다.
8. Docker에 대한 ACR 액세스를 설정하여 모든 Astra Control Center 이미지를 푸시/풀합니다.
9. 다음 스크립트를 입력하여 ACC 이미지를 이 레지스트리에 푸시합니다.

```
az acr login -n <AZ ACR URL/Location>
This script requires ACC manifest file and your Azure ACR location.
```

   - 예 *:

```
manifestfile=astra-control-center-<version>.manifest
AZ_ACR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRYY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

10. DNS 존 설정

#### NetApp Cloud Manager 구성

Cloud Manager를 사용하여 작업 영역을 만들고, Azure에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

Cloud Manager 설명서에 따라 다음 단계를 완료하십시오. 을 참조하십시오 "Azure에서 Cloud Manager 시작하기".

필요한 것

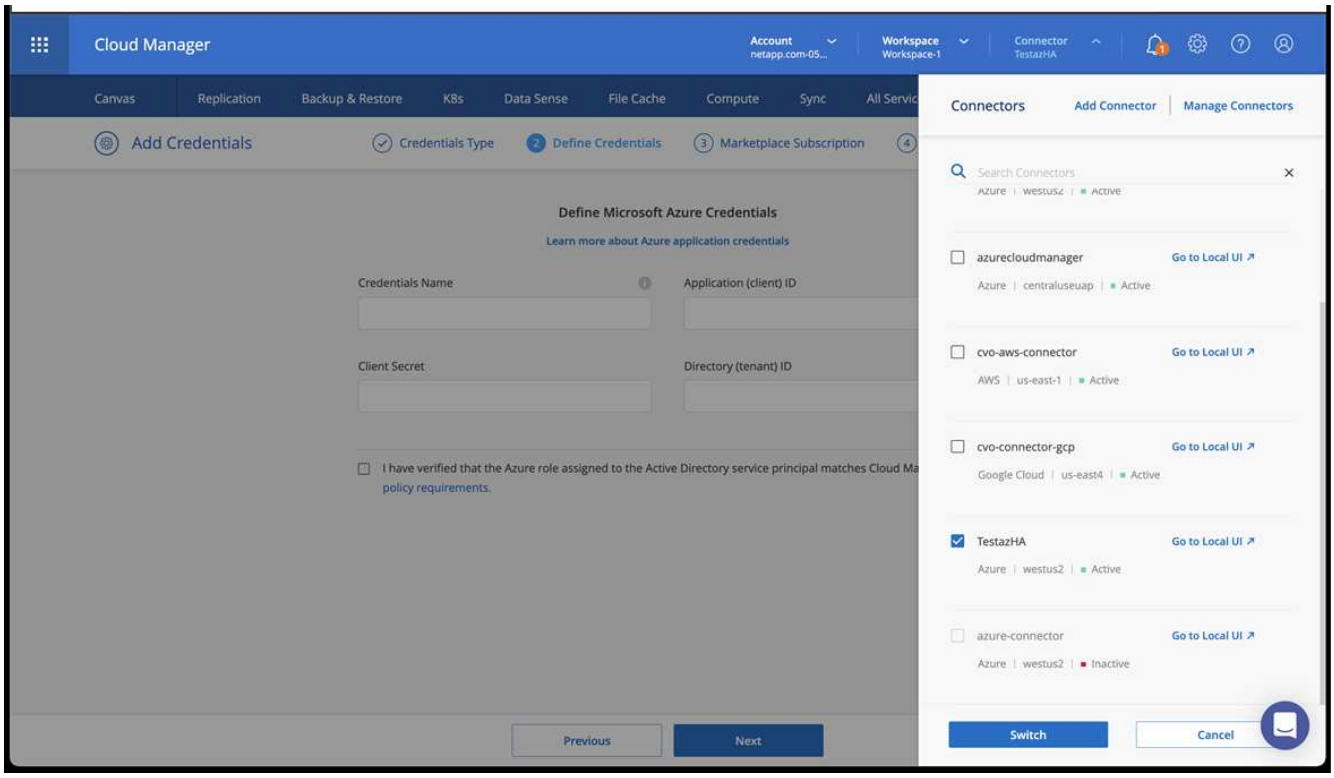
필요한 IAM 권한 및 역할을 사용하여 Azure 계정에 액세스합니다

단계

1. Cloud Manager에 자격 증명을 추가합니다.
2. Azure용 커넥터를 추가합니다. 을 참조하십시오 "Cloud Manager 정책".
   a. 공급자로 * Azure * 를 선택합니다.
   b. 애플리케이션 ID, 클라이언트 암호 및 디렉토리(테넌트) ID를 비롯한 Azure 자격 증명을 입력합니다.

을 참조하십시오 "Cloud Manager에서 Azure에 커넥터 만들기".

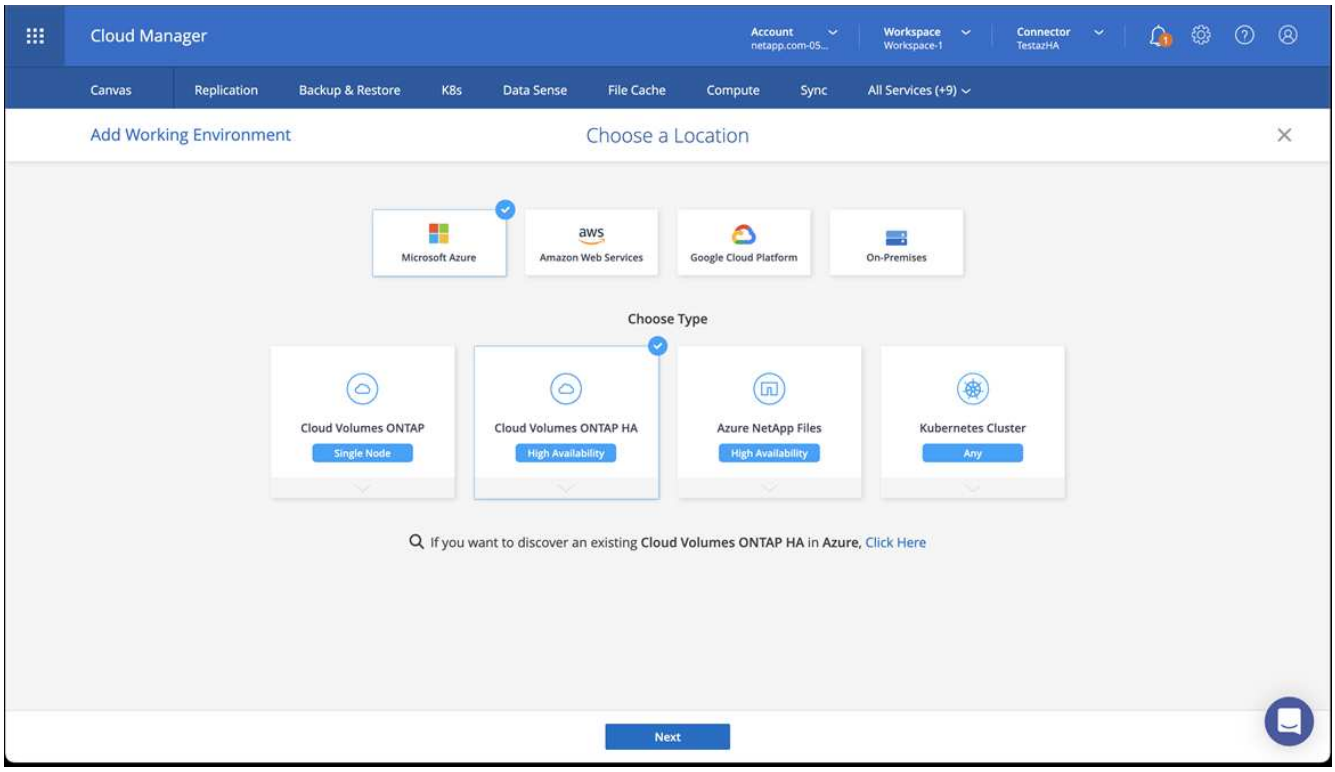
3. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.




4. 클라우드 환경을 위한 작업 환경을 구축합니다.

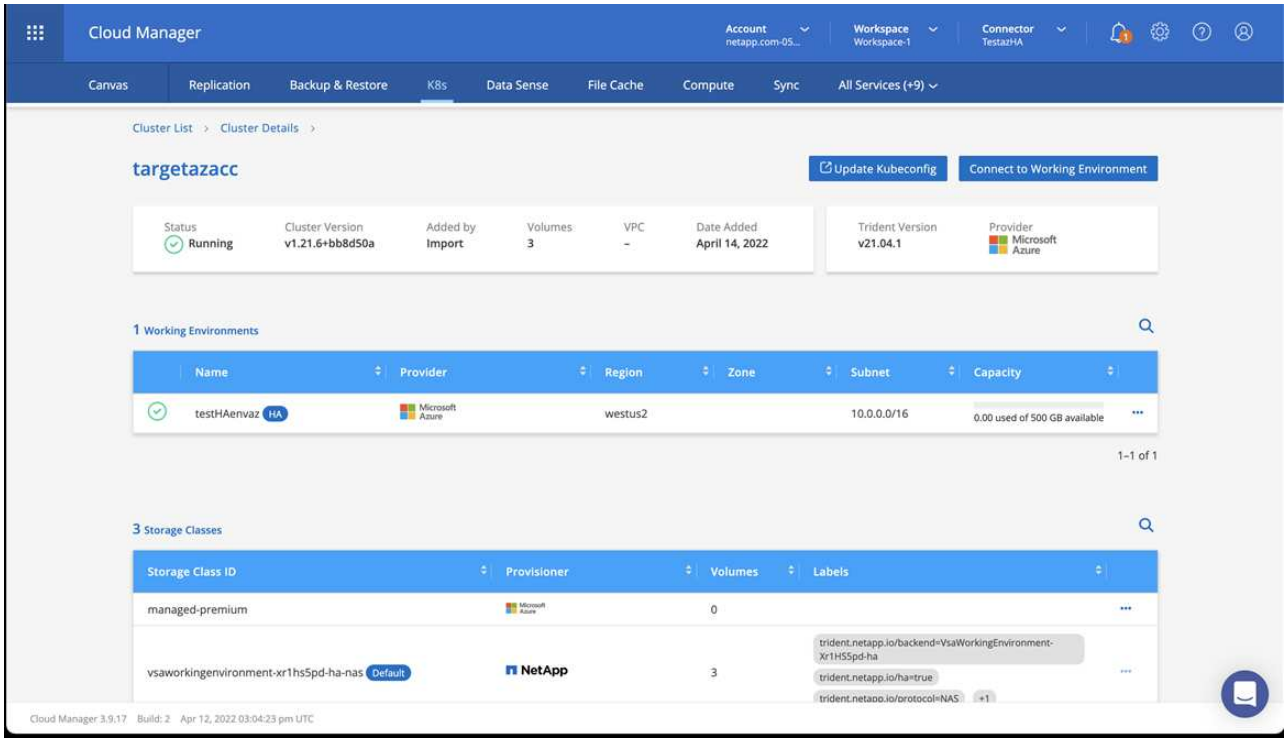
   a. 위치: "Microsoft Azure".

   b. "Cloud Volumes ONTAP HA"를 입력합니다.




5. OpenShift 클러스터를 가져옵니다. 클러스터가 방금 생성한 작업 환경에 연결됩니다.

   a. NetApp 클러스터 세부 정보를 보려면 * K8s * > * 클러스터 목록 * > * 클러스터 세부 정보 * 를 선택합니다.




   b. 오른쪽 위 모서리에서 Trident 버전을 확인합니다.

   c. NetApp을 공급자 로 보여주는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 참조하십시오.

이렇게 하면 Red Hat OpenShift 클러스터를 가져오고 기본 스토리지 클래스를 할당합니다. 스토리지 클래스를

선택합니다. Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

6. 이 Cloud Volumes ONTAP 배포에서 모든 영구 볼륨 및 볼륨을 기록해 둡니다.
7. Cloud Volumes ONTAP는 단일 노드 또는 고가용성으로 작동할 수 있습니다. HA가 활성화된 경우 Azure에서 실행 중인 HA 상태와 노드 배포 상태를 확인하십시오.

**Astra Control Center**를 설치하고 구성합니다

Astra Control Center를 표준으로 설치합니다 "설치 지침".

Astra Control Center를 사용하여 Azure 버킷을 추가합니다. 을 참조하십시오 "Astra Control Center를 설정하고 버킷을 추가합니다".

# Astra Control Center를 설정합니다

Astra Control Center는 스토리지 백엔드로 ONTAP 및 Astra 데이터 저장소를 지원 및 모니터링합니다. Astra Control Center를 설치하고, UI에 로그인하고, 암호를 변경하면 라이센스를 설정하고, 클러스터를 추가하고, 스토리지를 관리하고, 버킷을 추가할 수 있습니다.

작업

- Astra Control Center에 대한 라이센스를 추가합니다
- 클러스터 추가
- 스토리지 백엔드를 추가합니다
- 버킷을 추가합니다

## Astra Control Center에 대한 라이센스를 추가합니다

UI 또는 를 사용하여 새 라이센스를 추가할 수 있습니다 "API를 참조하십시오" Astra Control Center의 모든 기능을 활용할 수 있습니다. 라이센스가 없으면 Astra Control Center의 사용은 사용자 관리 및 새 클러스터 추가로 제한됩니다.

라이선스 계산 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

> 기존 평가판 또는 전체 라이센스를 업데이트하려면 을 참조하십시오 "기존 라이센스를 업데이트합니다".

Astra Control Center 라이센스는 Kubernetes CPU 장치를 사용하여 CPU 리소스를 측정합니다. 라이센스는 모든 관리되는 Kubernetes 클러스터의 작업자 노드에 할당된 CPU 리소스를 고려해야 합니다. 라이센스를 추가하기 전에 에서 라이센스 파일(NLF)을 얻어야 합니다 "NetApp Support 사이트".

또한 평가판 라이센스가 있는 Astra Control Center를 사용하여 라이센스를 다운로드한 날짜로부터 90일 동안 Astra Control Center를 사용할 수 있습니다. 등록하면 무료 평가판을 사용할 수 있습니다 "여기".

> 설치가 라이센스 CPU 유닛 수를 초과하여 증가할 경우, Astra Control Center를 통해 새 애플리케이션을 관리할 수 없습니다. 용량이 초과되면 경고가 표시됩니다.

필요한 것

에서 Astra Control Center를 다운로드한 경우 "NetApp Support 사이트"또한 NetApp 라이센스 파일(NLF)도 다운로드했습니다. 이 라이센스 파일에 대한 액세스 권한이 있는지 확인하십시오.

단계

1. Astra Control Center UI에 로그인합니다.
2. 계정 * > * 라이센스 * 를 선택합니다.
3. 라이센스 추가 * 를 선택합니다.
4. 다운로드한 라이센스 파일(NLF)으로 이동합니다.
5. 라이센스 추가 * 를 선택합니다.

계정 * > * 라이센스 * 페이지에는 라이센스 정보, 만료 날짜, 라이센스 일련 번호, 계정 ID 및 사용된 CPU 단위가 표시됩니다.

평가 라이센스가 있는 경우 ASUP를 보내지 않을 경우 Astra Control Center에 장애가 발생할 경우 데이터 손실을 방지하기 위해 계정 ID를 저장해야 합니다.

## 클러스터 추가

앱 관리를 시작하려면 Kubernetes 클러스터를 추가하고 이를 컴퓨팅 리소스로 관리합니다. Kubernetes 애플리케이션을 검색하려면 Astra Control Center용 클러스터를 추가해야 합니다. Astra Data Store의 경우, Astra Data Store에서 프로비저닝한 볼륨을 사용하는 애플리케이션이 포함된 Kubernetes 앱 클러스터를 추가하려고 합니다.

관리를 위해 Astra Control Center에 다른 클러스터를 추가하기 전에 먼저 Astra Control Center에서 클러스터를 관리하는 것이 좋습니다. 메트릭 및 문제 해결을 위해 Kubemetrics 데이터 및 클러스터 관련 데이터를 전송하려면 관리 중인 초기 클러스터가 필요합니다. 클러스터 추가 * 기능을 사용하여 Astra Control Center로 클러스터를 관리할 수 있습니다.

Astra Control이 클러스터를 관리할 때 클러스터의 기본 스토리지 클래스를 추적합니다. 를 사용하여 스토리지 클래스를 변경하는 경우 `kubectl` 명령, Astra Control은 변경 사항을 되돌립니다. Astra Control에서 관리하는 클러스터의 기본 스토리지 클래스를 변경하려면 다음 방법 중 하나를 사용하십시오.

- Astra Control API를 사용합니다 `PUT /managedClusters` 를 사용하여 다른 기본 스토리지 클래스를 할당합니다 `DefaultStorageClass` 매개 변수.
- Astra Control 웹 UI를 사용하여 다른 기본 스토리지 클래스를 할당합니다. 을 참조하십시오 기본 스토리지 클래스를 변경합니다.

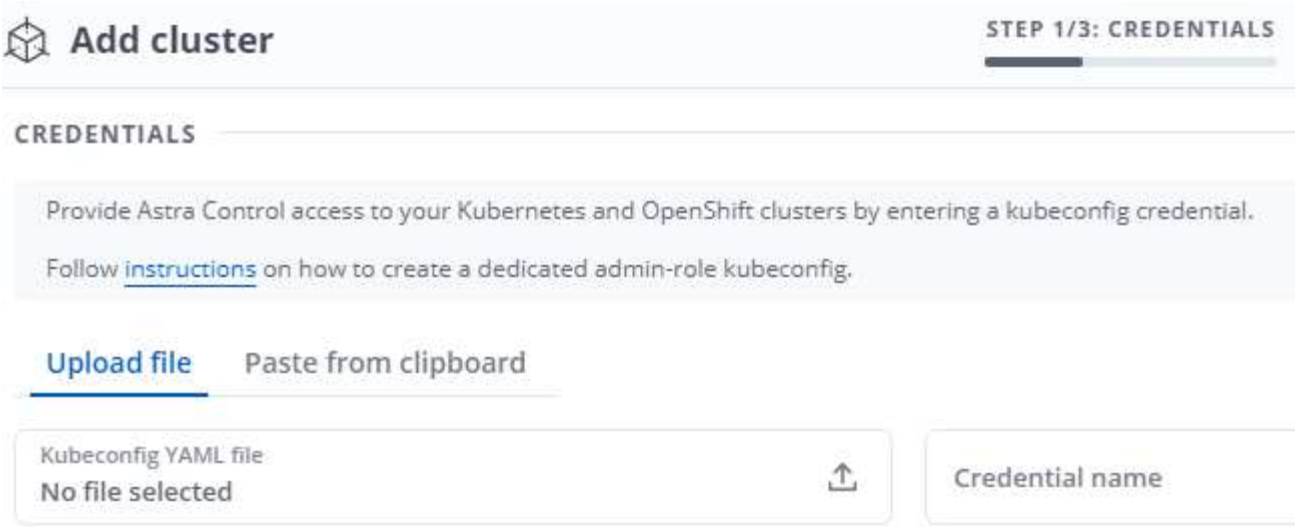
필요한 것

- 클러스터를 추가하기 전에 필요한 를 검토 및 수행합니다 "선행 작업".

단계

1. Astra Control Center UI의 * Dashboard * 에서 Clusters 섹션에서 * Add * 를 선택합니다.
2. 열리는 * Add Cluster * (클러스터 추가 *) 창에서 를 업로드합니다 `kubeconfig.yaml` 의 내용을 파일 또는 붙여 넣습니다 `kubeconfig.yaml` 파일.

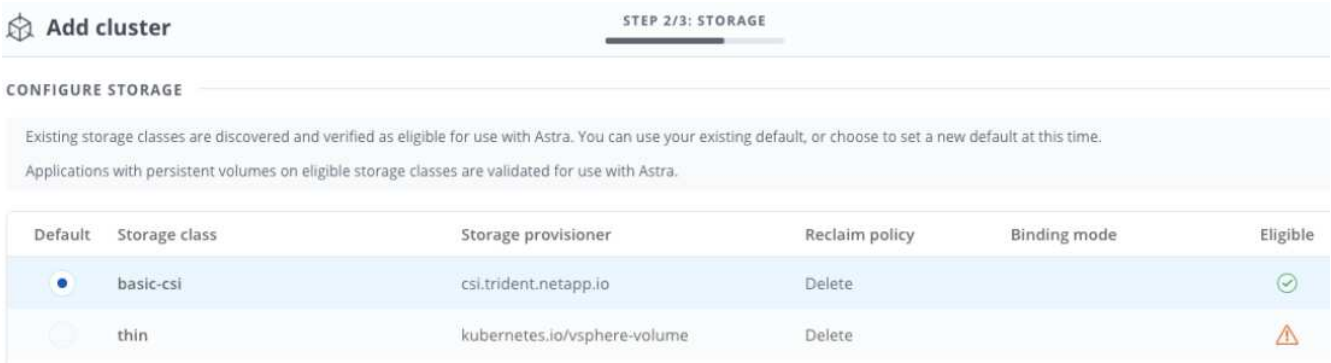
   를 클릭합니다 `kubeconfig.yaml` 파일에는 클러스터 자격 증명 1개에 대한 * 만 포함되어야 합니다 *.




> 직접 만드는 경우 `kubeconfig` 파일에서 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" 을 참조하십시오 `kubeconfig` 파일.

3. 자격 증명 이름을 제공하십시오. 기본적으로 자격 증명 이름은 클러스터 이름으로 자동 채워집니다.
4. 스토리지 구성 * 을 선택합니다.
5. 이 Kubernetes 클러스터에 사용할 스토리지 클래스를 선택하고 * Review * 를 선택하십시오.

> ONTAP 스토리지 또는 Astra 데이터 저장소에서 지원하는 Trident 스토리지 클래스를 선택해야 합니다.




6. 정보를 검토하고 모든 내용이 양호하면 * 클러스터 추가 * 를 선택합니다.

결과

클러스터가 * 검색 * 상태로 전환되고 * 실행 * 으로 변경됩니다. Kubernetes 클러스터를 성공적으로 추가했으며 현재 Astra Control Center에서 관리하고 있습니다.

> Astra Control Center에서 관리할 클러스터를 추가한 후 모니터링 연산자를 구축하는 데 몇 분이 걸릴 수 있습니다. 그 전까지는 알림 아이콘이 빨간색으로 바뀌고 * 모니터링 에이전트 상태 확인 실패 * 이벤트를 기록합니다. Astra Control Center가 올바른 상태를 획득하면 문제가 해결되므로 이 문제를 무시할 수 있습니다. 몇 분 이내에 문제가 해결되지 않으면 클러스터로 이동하여 를 실행합니다 `oc get pods -n netapp-monitoring` 시작점으로 사용됩니다. 문제를 디버깅하려면 모니터링 운영자 로그를 확인해야 합니다.

## 스토리지 백엔드를 추가합니다

Astra Control에서 리소스를 관리할 수 있도록 스토리지 백엔드를 추가할 수 있습니다. 관리되는 클러스터에 스토리지 백엔드를 구축하거나 기존 스토리지 백엔드를 사용할 수 있습니다.

Astra Control에서 스토리지 클러스터를 스토리지 백엔드로 관리하면 PVS(영구적 볼륨)와 스토리지 백엔드 간의 연결 및 추가 스토리지 메트릭을 얻을 수 있습니다.

기존 **Astra Data Store** 구축에 필요한 사항

- Kubernetes 앱 클러스터와 기본 컴퓨팅 클러스터가 추가되었습니다.

> Astra Data Store용 Kubernetes 앱 클러스터를 추가하고 Astra Control에서 관리하는 경우 클러스터가 로 표시됩니다 `unmanaged` 을 클릭합니다. 다음으로 Astra Data Store가 포함된 컴퓨팅 클러스터를 추가하고 Kubernetes 애플리케이션 클러스터를 포함해야 합니다. UI의 * backends * 에서 이 작업을 수행할 수 있습니다. 클러스터의 Actions 메뉴를 선택하고 를 선택합니다 `Manage`, 및 "클러스터를 추가합니다". 클러스터 상태 후 `unmanaged` Kubernetes 클러스터 이름이 변경되면 백엔드를 추가하는 작업을 계속 진행할 수 있습니다.

새로운 **Astra Data Store** 구축에 필요한 사항

- 있습니다 "배포하려는 설치 번들 버전을 업로드했습니다" Astra Control에 액세스할 수 있는 위치
- 배포에 사용할 Kubernetes 클러스터를 추가했습니다.
- 을(를) 업로드했습니다 Astra Data Store 라이센스 Astra Control에 액세스할 수 있는 위치에 배포할 수 있습니다.

옵션

- 스토리지 리소스 구축
- 기존 스토리지 백엔드를 사용합니다

스토리지 리소스 구축

새로운 Astra Data Store를 구축하고 관련 스토리지 백엔드를 관리할 수 있습니다.

단계

1. 대시보드 또는 백엔드 메뉴에서 이동합니다.
   - 대시보드 * 에서: 리소스 요약의 스토리지 백엔드 창에서 링크를 선택하고 백엔드 섹션에서 * 추가 * 를 선택합니다.
   - 시작 * 백엔드 *:
     i. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
     ii. 추가 * 를 선택합니다.
2. 배포 * 탭에서 * Astra Data Store * 배포 옵션을 선택합니다.
3. 배포할 Astra Data Store 패키지를 선택합니다.
   a. Astra Data Store 애플리케이션의 이름을 입력합니다.
   b. 배포할 Astra Data Store의 버전을 선택합니다.

> 배포하려는 버전을 아직 업로드하지 않은 경우 * 패키지 추가 * 옵션을 사용하거나 마법사를 종료하고 를 사용할 수 있습니다 "패키지 관리" 를 눌러 설치 번들을 업로드합니다.

4. 이전에 업로드한 Astra Data Store 라이센스를 선택하거나 * Add license * 옵션을 사용하여 응용 프로그램에 사용할 라이센스를 업로드합니다.

> 모든 권한이 있는 Astra Data Store 라이센스가 Kubernetes 클러스터와 연결되어 있으며, 이와 관련된 클러스터가 자동으로 표시됩니다. 관리되는 클러스터가 없는 경우 * 클러스터 추가 * 옵션을 선택하여 Astra Control 관리에 클러스터를 추가할 수 있습니다. Astra Data Store 라이센스의 경우 라이센스와 클러스터 간에 연결이 되지 않은 경우 마법사의 다음 페이지에서 이 연결을 정의할 수 있습니다.

5. Kubernetes 클러스터를 Astra Control 관리에 추가하지 않은 경우 * Kubernetes 클러스터 * 페이지에서 추가해야 합니다. 목록에서 기존 클러스터를 선택하거나 * 기본 클러스터 추가 * 를 선택하여 Astra Control 관리에 클러스터를 추가합니다.
6. Astra Data Store에 리소스를 제공할 Kubernetes 클러스터의 템플릿 크기를 선택합니다. 다음 중 하나를 선택할 수 있습니다.
   - 선택하십시오 `Recommended Kubernetes worker node requirements`에서 라이센스에 허용되는 내용에 따라 큰 서식 파일에서 작은 서식 파일을 선택합니다.
   - 선택하십시오 `Custom Kubernetes worker node requirements`에서 각 클러스터 노드에 대해 원하는 코어 수와 총 메모리를 선택합니다. 또한 코어 및 메모리에 대한 선택 기준을 충족하는 클러스터에 있는 노드의 수를 표시할 수도 있습니다.

> 템플릿을 선택할 때 더 큰 워크로드를 위해 더 많은 메모리와 코어를 가진 더 큰 노드를 선택하거나 더 작은 워크로드의 경우 더 많은 노드를 선택합니다. 라이센스에 허용되는 내용에 따라 템플릿을 선택해야 합니다. 권장되는 각 템플릿 옵션은 각 노드의 메모리 및 코어, 용량에 대한 템플릿 패턴을 충족하는 적합한 노드 수를 제안합니다.

7. 노드 구성:
   a. 노드 레이블을 추가하여 이 Astra Data Store 클러스터를 지원하는 작업자 노드 풀을 식별합니다.

   > 구축 또는 구축을 시작하기 전에 Astra Data Store 구축에 사용할 클러스터의 각 개별 노드에 레이블을 추가해야 합니다.

   b. 노드당 용량(GiB)을 수동으로 구성하거나 허용되는 최대 노드 용량을 선택합니다.
   c. 클러스터에서 허용되는 최대 노드 수를 구성하거나 클러스터에서 최대 노드 수를 허용합니다.
8. (Astra Data Store 전체 라이센스만 해당) 보호 도메인에 사용할 레이블의 키를 입력합니다.

> 각 노드에 대해 키에 대한 고유 레이블을 3개 이상 생성합니다. 예를 들어, 키가 인 경우 `astra.datastore.protection.domain`다음과 같은 레이블을 만들 수 있습니다. `astra.datastore.protection.domain=domain1`,`astra.datastore.protection.domain=domain2`, 및 `astra.datastore.protection.domain=domain3`.

9. 관리 네트워크 구성:
   a. 작업자 노드 IP 주소와 동일한 서브넷에 있는 Astra Data Store 내부 관리에 대한 관리 IP 주소를 입력합니다.
   b. 관리 및 데이터 네트워크 모두에 동일한 NIC를 사용하거나 별도로 구성합니다.

c. 스토리지 액세스를 위한 데이터 네트워크 IP 주소 풀, 서브넷 마스크 및 게이트웨이를 입력합니다.

10. 구성을 검토하고 * deploy * 를 선택하여 설치를 시작합니다.

결과

설치가 완료되면 백엔드가 에 나타납니다 `available` 활성 성능 정보와 함께 백엔드 목록의 상태입니다.

> 백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

기존 스토리지 백엔드를 사용합니다

검색된 ONTAP 또는 Astra Data Store 스토리지 백엔드를 Astra Control Center 관리 센터에 가져올 수 있습니다.

단계

1. 대시보드 또는 백엔드 메뉴에서 이동합니다.
   - 대시보드 * 에서: 리소스 요약의 스토리지 백엔드 창에서 링크를 선택하고 백엔드 섹션에서 * 추가 * 를 선택합니다.
   - 시작 * 백엔드 *:
     i. 왼쪽 탐색 영역에서 * backends * 를 선택합니다.
     ii. 관리되는 클러스터에서 검색된 백엔드에서 * 관리 * 를 선택하거나 * 추가 * 를 선택하여 기존 백엔드를 추가로 관리합니다.
2. 기존 * 사용 탭을 선택합니다.
3. 백엔드 유형에 따라 다음 중 하나를 수행합니다.
   - * Astra 데이터 저장소 *:
     i. Astra Data Store * 를 선택합니다.
     ii. 관리되는 컴퓨팅 클러스터를 선택하고 * Next * 를 선택합니다.
     iii. 백엔드 세부 정보를 확인하고 * Add storage backend * 를 선택합니다.
   - * ONTAP *:
     i. ONTAP * 를 선택하고 * Next * 를 선택합니다.
     ii. ONTAP 클러스터 관리 IP 주소 및 관리 자격 증명을 입력합니다.

     > 여기에 자격 증명을 입력한 사용자에게는 가 있어야 합니다 `ontapi` ONTAP 클러스터의 ONTAP System Manager에서 활성화된 사용자 로그인 액세스 방법입니다. SnapMirror 복제를 사용하려는 경우 액세스 방법을 설정합니다 `ontapi` 및 `http` 두 ONTAP 클러스터 모두에 있는 사용자의 경우. 을 참조하십시오 "사용자 계정 관리" 를 참조하십시오.

     iii. Review * 를 선택합니다.
     iv. 백엔드 세부 정보를 확인하고 * Add storage backend * 를 선택합니다.

결과

백엔드가 에 나타납니다 `available` 목록의 상태로 요약 정보를 표시합니다.

> 백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

## 버킷을 추가합니다

애플리케이션과 영구 스토리지를 백업하려는 경우나 클러스터 간에 애플리케이션을 클론 복제하려는 경우에는 오브젝트 저장소 버킷 공급자를 추가하는 것이 중요합니다. Astra Control은 이러한 백업 또는 클론을 정의한 오브젝트 저장소 버킷에 저장합니다.

버킷을 추가하면 Astra Control은 하나의 버킷을 기본 버킷 표시기로 표시합니다. 사용자가 만든 첫 번째 버킷이 기본 버킷이 됩니다.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제할 경우 버킷이 필요하지 않습니다.

다음 버킷 유형 중 하나를 사용하십시오.

- NetApp ONTAP S3
- NetApp StorageGRID S3
- 일반 S3

> AWS(Amazon Web Services) 및 GCP(Google Cloud Platform)는 일반 S3 버킷 유형을 사용합니다.

- Microsoft Azure를 참조하십시오

> Astra Control Center는 Amazon S3를 일반 S3 버킷 공급자로 지원하지만, Astra Control Center는 Amazon의 S3 지원을 주장하는 모든 오브젝트 저장소 공급업체를 지원하지 않을 수 있습니다.

- Microsoft Azure를 참조하십시오

Astra Control API를 사용하여 버킷을 추가하는 방법에 대한 지침은 를 참조하십시오 "Astra 자동화 및 API 정보".

단계

1. 왼쪽 탐색 영역에서 * Bucket * 을 선택합니다.
   a. 추가 * 를 선택합니다.
   b. 버킷 유형을 선택합니다.

   > 버킷을 추가할 때 올바른 버킷 공급자를 선택하고 해당 공급자에 적합한 자격 증명을 제공합니다. 예를 들어, UI에서 NetApp ONTAP S3를 유형으로 받아들이고 StorageGRID 자격 증명을 받아들이지만, 이 버킷을 사용한 이후의 모든 애플리케이션 백업 및 복원이 실패합니다.

   c. 새 버킷 이름을 생성하거나 기존 버킷 이름과 선택적 설명을 입력합니다.

   > 버킷 이름 및 설명은 백업을 생성할 때 나중에 선택할 수 있는 백업 위치로 나타납니다. 이 이름은 보호 정책 구성 중에도 표시됩니다.

   d. S3 엔드포인트의 이름 또는 IP 주소를 입력합니다.
   e. 이 버킷을 모든 백업의 기본 버킷으로 사용하려면 를 선택합니다 `Make this bucket the default bucket for this private cloud` 옵션을 선택합니다.

> 이 옵션은 사용자가 만든 첫 번째 버킷에는 나타나지 않습니다.

f. 를 추가하여 계속합니다 자격 증명 정보.

**S3** 액세스 자격 증명을 추가합니다

언제든지 S3 액세스 자격 증명을 추가할 수 있습니다.

단계

1. Bucket 대화상자에서 * Add * 또는 * Use Existing * 탭을 선택합니다.
   a. Astra Control의 다른 자격 증명과 구별되는 자격 증명의 이름을 입력합니다.
   b. 클립보드의 내용을 붙여 넣어 액세스 ID와 비밀 키를 입력합니다.

## 기본 스토리지 클래스를 변경합니다

클러스터의 기본 스토리지 클래스를 변경할 수 있습니다.

단계

1. Astra Control Center 웹 UI에서 * Clusters * 를 선택합니다.
2. 클러스터 * 페이지에서 변경할 클러스터를 선택합니다.
3. Storage * 탭을 선택합니다.
4. 스토리지 클래스 * 범주를 선택합니다.
5. 기본값으로 설정할 스토리지 클래스에 대해 * Actions * 메뉴를 선택합니다.
6. Set as default * 를 선택합니다.

## 다음 단계

Astra Control Center에 로그인하고 클러스터를 추가했으므로 이제 Astra Control Center의 애플리케이션 데이터 관리 기능을 사용할 준비가 되었습니다.

- "사용자 관리"
- "앱 관리를 시작합니다"
- "앱 보호"
- "앱 클론 복제"
- "알림을 관리합니다"
- "Cloud Insights에 연결합니다"
- "사용자 지정 TLS 인증서를 추가합니다"

## 자세한 내용을 확인하십시오

- "Astra Control API를 사용합니다"
- "알려진 문제"

## 클러스터 추가를 위한 사전 요구사항

클러스터를 추가하기 전에 사전 요구 조건이 충족되는지 확인해야 합니다. 또한 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인해야 합니다.

### 클러스터를 추가하기 전에 필요한 사항

클러스터가 에 나와 있는 요구 사항을 충족하는지 확인합니다 "애플리케이션 클러스터 요구사항".

> 관리되는 컴퓨팅 리소스로 두 번째 OpenShift 4.6, 4.7 또는 4.8 클러스터를 추가하려는 경우 Astra Trident Volume Snapshot 기능이 활성화되어 있는지 확인해야 합니다. 공식 Astra Trident를 참조하십시오 "지침" Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트합니다.

- A로 구성된 Astra Trident StorageClasses "지원되는 스토리지 백엔드" (모든 유형의 클러스터에 필요)
- Astra Control Center를 사용하여 앱을 백업 및 복원하기 위해 백업 ONTAP 시스템에 설정된 고급 사용자 및 사용자 ID입니다. ONTAP 명령줄에서 다음 명령을 실행합니다.
  `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser sysm --anon 65534`
- Astra Trident `volumesnapshotclass` 관리자가 정의한 개체입니다. Astra Trident를 참조하십시오 "지침" Astra Trident를 사용하여 볼륨 스냅샷을 활성화하고 테스트합니다.
- Kubernetes 클러스터에 대해 단일 기본 스토리지 클래스만 정의되어 있는지 확인하십시오.

### 자격 검사를 실행합니다

다음 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인합니다.

단계

1. Trident 버전을 확인합니다.

   ```
   kubectl get tridentversions -n trident
   ```

   Trident가 있으면 다음과 유사한 출력이 표시됩니다.

   ```
   NAME       VERSION
   trident    21.04.0
   ```

   Trident가 없으면 다음과 유사한 출력이 표시됩니다.

   ```
   error: the server doesn't have a resource type "tridentversions"
   ```

   > Trident가 설치되지 않았거나 설치된 버전이 최신 버전이 아닌 경우 계속하기 전에 Trident의 최신 버전을 설치해야 합니다. 를 참조하십시오 "Trident 문서" 를 참조하십시오.

2. 스토리지 클래스가 지원되는 Trident 드라이버를 사용하고 있는지 확인합니다. 공급자 이름은 이어야 합니다

`csi.trident.netapp.io`. 다음 예를 참조하십시오.

```
kubectl get sc
NAME                    PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)    csi.trident.netapp.io          Delete
Immediate           true                   5d23h
thin                    kubernetes.io/vsphere-volume   Delete
Immediate           false                  6d
```

관리자 역할 **kubecononfig**를 생성합니다

단계를 수행하기 전에 시스템에 다음 사항이 있는지 확인하십시오.

- `kubectl` V1.19 이상이 설치되었습니다
- 활성 컨텍스트에 대한 클러스터 관리자 권한이 있는 활성 kubecononfig

단계

1. 다음과 같이 서비스 계정을 생성합니다.

   a. 라는 서비스 계정 파일을 생성합니다 `astracontrol-service-account.yaml`.

      필요에 따라 이름 및 네임스페이스를 조정합니다. 여기에서 변경한 경우 다음 단계에서 동일한 변경 사항을 적용해야 합니다.

```
<strong>astracontrol-service-account.yaml</strong>
```

+

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

   a. 서비스 계정 적용:

```
kubectl apply -f astracontrol-service-account.yaml
```

2. (선택 사항) 클러스터에서 권한이 있는 POD 생성을 허용하지 않거나 POD 컨테이너 내의 프로세스가 루트 사용자로 실행되도록 허용하지 않는 제한적인 POD 보안 정책을 사용하는 경우 Astra Control에서 POD를 생성 및 관리할 수 있도록 클러스터에 대한 사용자 지정 POD 보안 정책을 생성합니다. 자세한 내용은 을 참조하십시오 "사용자 지정 POD 보안 정책을 생성합니다".

3. 다음과 같이 클러스터 관리자 권한을 부여합니다.

   a. 을 생성합니다 `ClusterRoleBinding` 파일을 호출했습니다 `astracontrol-clusterrolebinding.yaml`.

      필요에 따라 서비스 계정을 생성할 때 수정된 모든 이름과 네임스페이스를 조정합니다.

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

+

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

   a. 클러스터 역할 바인딩을 적용합니다.

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

4. 교체 서비스 계정 암호를 나열합니다 `<context>` 올바른 설치 상황:

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

출력의 끝은 다음과 유사합니다.

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-vhz87"},
{ "name": "astracontrol-service-account-token-r59kr"}
]
```

의 각 요소에 대한 인덱스입니다 `secrets` 어레이는 0으로 시작합니다. 위의 예에서 의 인덱스입니다 `astracontrol-service-account-dockercfg-vhz87` 는 0이고 의 인덱스입니다 `astracontrol-`

`service-account-token-r59kr` 1입니다. 출력에서 "token"이라는 단어가 포함된 서비스 계정 이름의 인덱스를 기록해 둡니다.

5. 다음과 같이 kubecononfig를 생성합니다.

   a. 을 생성합니다 `create-kubeconfig.sh` 파일. 대치 `TOKEN_INDEX` 다음 스크립트의 시작 부분에 올바른 값이 있습니다.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp
```

```
# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

```
source create-kubeconfig.sh
```

6. (* 선택 사항 *) kubeconfig의 이름을 클러스터의 의미 있는 이름으로 바꿉니다. 클러스터 자격 증명을 보호합니다.

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

다음 단계

이제 필수 구성 요소가 충족되었는지 확인했으므로 이제 수행할 준비가 되었습니다 "클러스터를 추가합니다".

자세한 내용을 확인하십시오

- "Trident 문서"
- "Astra Control API를 사용합니다"

## 사용자 지정 **TLS** 인증서를 추가합니다

기존의 자체 서명된 TLS 인증서를 제거하고 CA(인증 기관)에서 서명한 TLS 인증서로 바꿀 수 있습니다.

필요한 것

- Astra Control Center가 설치된 Kubernetes 클러스터
- 실행할 클러스터의 명령 셸에 대한 관리 액세스 `kubectl` 명령
- CA의 개인 키 및 인증서 파일

### 자체 서명된 인증서를 제거합니다

기존의 자체 서명된 TLS 인증서를 제거합니다.

1. SSH를 사용하여 관리 사용자로 Astra Control Center를 호스팅하는 Kubernetes 클러스터에 로그인합니다.
2. 다음 명령을 사용하여 현재 인증서와 연결된 TLS 암호를 찾아 바꿉니다 `<ACC-deployment-namespace>` Astra Control Center 배포 네임스페이스 사용:

```
kubectl get certificate -n <ACC-deployment-namespace>
```

3. 다음 명령을 사용하여 현재 설치된 암호 및 인증서를 삭제합니다.

```
kubectl delete cert cert-manager-certificates -n <ACC-deployment-
namespace>
kubectl delete secret secure-testing-cert -n <ACC-deployment-namespace>
```

### 새 인증서를 추가합니다

CA에서 서명한 새 TLS 인증서를 추가합니다.

1. 다음 명령을 사용하여 CA의 개인 키 및 인증서 파일로 새 TLS 암호를 만들고 대괄호 <>의 인수를 적절한 정보로 바꿉니다.

```
kubectl create secret tls <secret-name> --key <private-key-filename>
--cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. 다음 명령 및 예제를 사용하여 클러스터 CRD(Custom Resource Definition) 파일을 편집하고 를 변경합니다 `spec.selfSigned` 값을 로 설정합니다 `spec.ca.secretName` 앞에서 만든 TLS 암호를 확인하려면 다음을 수행하십시오.

```
kubectl edit clusterissuers.cert-manager.io/cert-manager-certificates -n
<ACC-deployment-namespace>
....

#spec:
#  selfSigned: {}

spec:
  ca:
    secretName: <secret-name>
```

3. 다음 명령 및 예제 출력을 사용하여 변경 사항이 올바른지, 클러스터가 인증서를 교체할 준비가 되었는지 확인합니다 `<ACC-deployment-namespace>` Astra Control Center 배포 네임스페이스 사용:

```
kubectl describe clusterissuers.cert-manager.io/cert-manager-
certificates -n <ACC-deployment-namespace>
....

Status:
  Conditions:
    Last Transition Time:  2021-07-01T23:50:27Z
    Message:               Signing CA verified
    Reason:                KeyPairVerified
    Status:                True
    Type:                  Ready
Events:                    <none>
```

4. 를 생성합니다 `certificate.yaml` 다음 예제를 사용하는 파일 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다.

```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <certificate-name>
  namespace: <ACC-deployment-namespace>
spec:
  secretName: <certificate-secret-name>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  - <astra.dnsname.example.com> #Replace with the correct Astra Control
Center DNS address
  issuerRef:
    kind: ClusterIssuer
    name: cert-manager-certificates
```

5. 다음 명령을 사용하여 인증서를 생성합니다.

```
kubectl apply -f certificate.yaml
```

6. 다음 명령 및 예제 출력을 사용하여 인증서가 올바르게 만들어졌는지, 그리고 생성 중에 지정한 인수(예: 이름, 기간, 갱신 기한 및 DNS 이름)를 사용하여 확인합니다.

```
kubectl describe certificate -n <ACC-deployment-namespace>
....

Spec:
  Dns Names:
    astra.example.com
  Duration:  125h0m0s
  Issuer Ref:
    Kind:        ClusterIssuer
    Name:        cert-manager-certificates
  Renew Before:  61h0m0s
  Secret Name:   <certificate-secret-name>
Status:
  Conditions:
    Last Transition Time:  2021-07-02T00:45:41Z
    Message:               Certificate is up to date and has not expired
    Reason:                Ready
    Status:                True
    Type:                  Ready
  Not After:               2021-07-07T05:45:41Z
  Not Before:              2021-07-02T00:45:41Z
  Renewal Time:            2021-07-04T16:45:41Z
  Revision:                1
Events:                    <none>
```

7. 다음 명령 및 예제를 사용하여 새 인증서 암호를 가리키도록 수신 CRD TLS 옵션을 편집합니다. 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다.

```
kubectl edit ingressroutes.traefik.containo.us -n <ACC-deployment-
namespace>
....

# tls:
#    options:
#      name: default
#    secretName: secure-testing-cert
#    store:
#      name: default

 tls:
    options:
      name: default
    secretName: <certificate-secret-name>
    store:
      name: default
```

8. 웹 브라우저를 사용하여 Astra Control Center의 배포 IP 주소로 이동합니다.
9. 인증서 세부 정보가 설치한 인증서의 세부 정보와 일치하는지 확인합니다.
10. 인증서를 내보내고 결과를 웹 브라우저의 인증서 관리자로 가져옵니다.

## 사용자 지정 **POD** 보안 정책을 생성합니다

Astra Control은 관리하는 클러스터에서 Kubernetes Pod를 생성 및 관리해야 합니다. 클러스터에서 권한이 있는 POD 생성을 허용하지 않거나 POD 컨테이너 내의 프로세스가 루트 사용자로 실행되도록 허용하는 제한적인 POD 보안 정책을 사용하는 경우, Astra Control에서 이러한 POD를 생성 및 관리할 수 있도록 덜 제한적인 POD 보안 정책을 만들어야 합니다.

단계

1. 기본값보다 덜 제한적인 클러스터에 대한 POD 보안 정책을 생성하여 파일에 저장합니다. 예를 들면 다음과 같습니다.

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: astracontrol
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
```

2. POD 보안 정책에 대한 새 역할을 생성합니다.

```bash
kubectl-admin create role psp:astracontrol \
    --verb=use \
    --resource=podsecuritypolicy \
    --resource-name=astracontrol
```

3. 새 역할을 서비스 계정에 바인딩합니다.

```bash
kubectl-admin create rolebinding default:psp:astracontrol \
    --role=psp:astracontrol \
    --serviceaccount=astracontrol-service-account:default
```

# Astra Control Center에 대한 질문과 대답

이 FAQ는 질문에 대한 간단한 답변을 찾는 경우에 도움이 될 수 있습니다.

## 개요

다음 섹션에서는 Astra Control Center를 사용할 때 나타날 수 있는 몇 가지 추가 질문에 대한 답변을 제공합니다. 자세한 내용은 astra.feedback@netapp.com 으로 문의하십시오

## Astra Control Center에 액세스할 수 있습니다

- Astra Control URL은 무엇입니까? *

Astra Control Center는 로컬 인증과 각 환경에 고유한 URL을 사용합니다.

URL의 경우 브라우저에서 Astra_control_center_min YAML 사용자 정의 리소스 정의(CRD) 파일(Astra Control Center 설치 시)의 spec.astraAddress 필드에 설정한 FQDN(정규화된 도메인 이름)을 입력합니다. 이메일은 Astra_control_center_min YAML CRD의 spec.email 필드에 설정한 값입니다.

## 라이센싱

- 평가판 라이센스를 사용하고 있습니다. 전체 라이센스로 변경하는 방법은 무엇입니까? *

NetApp 라이센스 파일(NLF)을 받아 전체 라이센스로 쉽게 변경할 수 있습니다.

- 단계 *
- 왼쪽 탐색 창에서 * 계정 * > * 라이센스 * 를 선택합니다.
- 라이선스 추가 * 를 선택합니다.
- 다운로드한 라이센스 파일을 찾아 * 추가 * 를 선택합니다.
- 평가판 라이센스를 사용하고 있습니다. 앱을 관리할 수 있습니까? *

예. 평가판 라이센스를 사용하여 관리 앱 기능을 테스트할 수 있습니다.

## Kubernetes 클러스터를 등록하는 중입니다

- Astra Control에 추가한 후 Kubernetes 클러스터에 작업자 노드를 추가해야 합니다. 어떻게 해야 합니까? *

새 작업자 노드를 기존 풀에 추가할 수 있습니다. 이러한 정보는 Astra Control에서 자동으로 발견됩니다. Astra Control에서 새 노드가 보이지 않으면 새 작업자 노드가 지원되는 이미지 유형을 실행하고 있는지 확인합니다. 을 사용하여 새 작업자 노드의 상태를 확인할 수도 있습니다 `kubectl get nodes` 명령.

- 클러스터를 올바르게 관리하려면 어떻게 해야 합니까? *
  1. "Astra Control에서 애플리케이션을 관리합니다".
  2. "Astra Control에서 클러스터 관리를 해제합니다".
- Astra Control에서 Kubernetes 클러스터를 제거한 후 애플리케이션과 데이터는 어떻게 됩니까? *

Astra Control에서 클러스터를 제거해도 클러스터의 구성(애플리케이션 및 영구 스토리지)은 변경되지 않습니다. Astra

Control 스냅샷 또는 해당 클러스터의 애플리케이션 백업을 복구할 수 없습니다. Astra Control에서 생성한 영구 스토리지 백업은 Astra Control 내에 남아 있지만 복구할 수 없습니다.

> 다른 방법을 통해 클러스터를 삭제하기 전에 항상 Astra Control에서 클러스터를 제거하십시오. Astra Control에서 관리하는 다른 도구를 사용하여 클러스터를 삭제하면 Astra Control 계정에 문제가 발생할 수 있습니다.

- 관리를 해제하면 NetApp Trident가 클러스터에서 자동으로 제거됩니까? * Astra Control Center에서 클러스터를 관리할 때 Trident가 클러스터에서 자동으로 제거되지 않습니다. Trident를 제거하려면 가 필요합니다 "Trident 문서의 다음 단계를 따릅니다".

## 응용 프로그램 관리

- Astra Control이 응용 프로그램을 배포할 수 있습니까? *

Astra Control은 애플리케이션을 배포하지 않습니다. 응용 프로그램은 Astra Control 외부에서 배포해야 합니다.

- Astra Control에서 관리를 중지한 후 응용 프로그램은 어떻게 됩니까? *

기존 백업 또는 스냅샷이 삭제됩니다. 애플리케이션과 데이터는 사용 가능한 상태로 유지됩니다. 관리되지 않는 응용 프로그램 또는 해당 응용 프로그램에 속한 백업 또는 스냅숏에는 데이터 관리 작업을 사용할 수 없습니다.

- Astra Control이 NetApp이 아닌 스토리지에 있는 애플리케이션을 관리할 수 있습니까? *

아니요 Astra Control은 NetApp이 아닌 스토리지를 사용하는 애플리케이션을 검색할 수 있지만, NetApp이 아닌 스토리지를 사용하는 애플리케이션은 관리할 수 없습니다.

"Astra Control 자체를 관리해야 하나요?" "아닙니다. Astra Control 자체는 "시스템 앱"이기 때문에 관리하지 말아야 합니다.

- 비정상적인 포드가 앱 관리에 영향을 미치나요? * 관리 애플리케이션에 상태가 불량한 포드가 있는 경우, Astra Control은 새 백업 및 클론을 생성할 수 없습니다.

## 데이터 관리 작업

- 내 계정에 생성하지 않은 스냅샷이 있습니다. 어디에서 왔습니까? *

일부 상황에서는 Astra Control이 백업, 클론 또는 복원 프로세스의 일부로 스냅샷을 자동으로 생성합니다.

- My Application은 여러 PVS를 사용합니다. Astra Control이 이러한 모든 PVC의 스냅샷 및 백업을 수행할까요? *

예. Astra Control의 애플리케이션에 대한 스냅샷 작업에는 애플리케이션의 PVC에 바인딩된 모든 PVS의 스냅샷이 포함됩니다.

- Astra Control에서 생성한 스냅샷을 다른 인터페이스 또는 객체 스토리지를 통해 직접 관리할 수 있습니까? *

아니요 Astra Control에서 생성한 스냅샷 및 백업은 Astra Control에서만 관리할 수 있습니다.

저작권 정보

Copyright © 2023 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

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

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

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.