시작하십시오
Astra Control Center

NetApp
August 11, 2025

목차

시작하십시오

Astra Control에 대해 알아보십시오

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리 솔루션으로, 상태 저장 애플리케이션의 운영을 단순화합니다. Kubernetes 워크로드를 손쉽게 보호, 백업, 복제, 마이그레이션하고 정상 작동하는 애플리케이션 클론을 즉시 생성할 수 있습니다.

피처

Astra Control은 Kubernetes 애플리케이션 데이터 라이프사이클 관리에 중요한 기능을 제공합니다.

- 영구 스토리지를 자동으로 관리합니다
- 애플리케이션 인식 필요 시 스냅샷과 백업을 생성합니다
- 정책 기반 스냅샷 및 백업 작업 자동화
- Kubernetes 클러스터 간에 애플리케이션 및 데이터를 마이그레이션합니다
- NetApp SnapMirror 기술(Astra Control Center)을 사용하여 원격 시스템에 애플리케이션 복제
- 스테이징 환경에서 운영 환경으로 애플리케이션 클론 생성
- 애플리케이션 상태 및 보호 상태를 시각화합니다
- 웹 UI 또는 API를 사용하여 백업 및 마이그레이션 워크플로우를 구현합니다

구축 모델

Astra Control은 두 가지 배포 모델로 제공됩니다.

- * Astra Control Service *: 여러 클라우드 공급자 환경의 Kubernetes 클러스터에 대한 애플리케이션 인식 데이터 관리 기능과 자가 관리 Kubernetes 클러스터를 제공하는 NetApp 관리 서비스입니다.
- * Astra Control Center *: 사내 환경에서 실행되는 Kubernetes 클러스터의 애플리케이션 인식 데이터 관리를 제공하는 자체 관리 소프트웨어입니다. 또한 NetApp Cloud Volumes ONTAP 스토리지 백엔드를 통해 여러 클라우드 공급자 환경에 Astra Control Center를 설치할 수 있습니다.

	Astra 제어 서비스	**Astra** 제어 센터
어떻게 제공됩니까**?**	NetApp에서 제공하는 완전 관리형 클라우드 서비스	소프트웨어를 다운로드, 설치 및 관리할 수 있습니다
어디에 호스팅됩니까**?**	NetApp에서 제공하는 다양한 퍼블릭 클라우드 지원	고유한 Kubernetes 클러스터
어떻게 업데이트됩니까**?**	NetApp에서 관리합니다	모든 업데이트를 관리합니다

	Astra 제어 서비스	**Astra** 제어 센터
지원되는 **Kubernetes** 배포는 무엇입니까**?**	• * 클라우드 공급자 * ◦ Amazon Web Services에서 직접 지원합니다 ▪ Amazon Elastic Kubernetes Service(EKS) ◦ Google 클라우드 ▪ Google Kubernetes Engine(GKE) ◦ Microsoft Azure를 참조하십시오 ▪ Azure Kubernetes 서비스(AKS) • * 자가 관리형 클러스터 * ◦ Kubernetes(업스트림) ◦ RKE(Rancher Kubernetes Engine) ◦ Red Hat OpenShift Container Platform • * 온프레미스 클러스터 * ◦ Red Hat OpenShift Container Platform 온프레미스	• Azure Stack HCI 기반 Azure Kubernetes Service • Google Anthos • Kubernetes(업스트림) • RKE(Rancher Kubernetes Engine) • Red Hat OpenShift Container Platform

	Astra 제어 서비스	**Astra** 제어 센터
지원되는 스토리지 백엔드는 무엇입니까?	• * 클라우드 공급자 * ◦ Amazon Web Services에서 직접 지원합니다 ▪ Amazon EBS ▪ NetApp ONTAP용 Amazon FSx ▪ "Cloud Volumes ONTAP" ◦ Google 클라우드 ▪ Google 영구 디스크 ▪ NetApp Cloud Volumes Service를 참조하십시오 ▪ "Cloud Volumes ONTAP" ◦ Microsoft Azure를 참조하십시오 ▪ Azure 관리 디스크 ▪ Azure NetApp Files ▪ "Cloud Volumes ONTAP" • * 자가 관리형 클러스터 * ◦ Amazon EBS ◦ Azure 관리 디스크 ◦ Google 영구 디스크 ◦ "Cloud Volumes ONTAP" ◦ NetApp MetroCluster ◦ "롱혼" • * 온프레미스 클러스터 * ◦ NetApp MetroCluster ◦ NetApp ONTAP AFF 및 FAS 시스템 ◦ NetApp ONTAP Select를 참조하십시오 ◦ "Cloud Volumes ONTAP" ◦ "롱혼"	• NetApp ONTAP AFF 및 FAS 시스템 • NetApp ONTAP Select를 참조하십시오 • "Cloud Volumes ONTAP" • "롱혼"

Astra Control Service의 작동 방식

Astra Control Service는 NetApp에서 관리하는 클라우드 서비스로, 항상 최신 기능을 사용하여 업데이트 가능합니다. 이 솔루션은 여러 구성 요소를 활용하여 애플리케이션 데이터 수명 주기 관리를 지원합니다.

높은 수준에서 Astra Control Service는 다음과 같이 작동합니다.

- 클라우드 공급자를 설정하고 Astra 계정에 등록하여 Astra Control Service를 시작할 수 있습니다.
  - GKE 클러스터의 경우 Astra Control Service가 사용합니다 "Google Cloud용 NetApp Cloud Volumes Service" 또는 Google 영구 디스크를 영구 볼륨의 스토리지 백엔드로 사용합니다.
  - AKS 클러스터의 경우 Astra Control Service가 사용합니다 "Azure NetApp Files" 또는 Azure 관리 디스크를 영구 볼륨의 스토리지 백엔드로 사용합니다.
  - Amazon EKS 클러스터의 경우 Astra Control Service가 사용합니다 "Amazon Elastic Block Store를 클릭합니다" 또는 "NetApp ONTAP용 Amazon FSx" 영구 볼륨의 스토리지 백엔드로 사용됩니다.
- 첫 번째 Kubernetes 컴퓨팅을 Astra Control Service에 추가합니다. 그러면 Astra Control Service에서 다음을 수행합니다.
  - 클라우드 공급자 계정에 백업 복사본이 저장되는 개체 저장소를 만듭니다.

    Azure에서 Astra Control Service는 Blob 컨테이너용 리소스 그룹, 스토리지 계정 및 키도 생성합니다.
  - 클러스터에 새 관리 역할 및 Kubernetes 서비스 계정을 생성합니다.
  - 이 새 관리자 역할을 사용하여 클러스터에 link../conceptions/architecture #Astra-control-components[Astra Control Provisioner]를 설치하고 하나 이상의 스토리지 클래스를 생성합니다.
  - NetApp 클라우드 서비스 스토리지 오퍼링을 스토리지 백엔드로 사용하는 경우 Astra Control Service는 Astra Control Provisioner를 사용하여 앱에 영구 볼륨을 프로비저닝합니다. Amazon EBS 또는 Azure 관리 디스크를 스토리지 백엔드로 사용하는 경우 공급자별 CSI 드라이버를 설치해야 합니다. 설치 지침은 에 나와 있습니다 "Amazon Web Services를 설정합니다" 및 "Azure 관리 디스크를 사용하여 Microsoft Azure를 설정합니다".
- 이제 앱을 클러스터에 추가할 수 있습니다. 영구 볼륨은 새로운 기본 스토리지 클래스에 프로비저닝됩니다.
- 그런 다음 Astra Control Service를 사용하여 이러한 애플리케이션을 관리하고 스냅샷, 백업 및 클론 생성을 시작합니다.

Astra Control의 무료 플랜을 사용하면 최대 10개의 네임스페이스를 계정에서 관리할 수 있습니다. 10개 이상의 항목을 관리하려는 경우 무료 요금에서 프리미엄 요금제로 업그레이드하여 청구서를 설정해야 합니다.

Astra Control Center의 작동 방식

Astra Control Center는 프라이빗 클라우드에서 로컬로 실행됩니다.

Astra Control Center는 ONTAP 스토리지 백엔드와 함께 Astra Control Provisioner 구성 스토리지 클래스를 통해 Kubernetes 클러스터를 지원합니다.

Astra Control Center에서 제한된(7일간 메트릭) 모니터링 및 원격 측정 기능을 사용할 수 있으며, 개방형 메트릭 엔드포인트를 통해 Kubernetes 기본 모니터링 툴(예: Prometheus 및 Grafana)로 내보낼 수도 있습니다.

Astra Control Center는 AutoSupport 및 Active IQ Digital Advisor(Digital Advisor라고도 함) 에코시스템에 완벽하게 통합되어 사용자 및 NetApp 지원에 문제 해결 및 사용 정보를 제공합니다.

90일 임베디드 평가판 라이센스를 사용하여 Astra Control Center를 사용해 볼 수 있습니다. Astra Control Center를 평가하는 동안 이메일과 커뮤니티 옵션을 통해 지원을 받을 수 있습니다. 또한 제품 내 지원 대시보드에서 Knowledgebase 문서 및 문서에 액세스할 수 있습니다.

Astra Control Center를 설치하고 사용하려면 반드시 충족해야 합니다 "요구 사항".

Astra Control Center는 다음과 같이 높은 수준에서 작동합니다.

- 현지 환경에 Astra Control Center를 설치합니다. 에 대해 자세히 알아보십시오 "Astra Control Center를 설치합니다".
- 다음과 같은 몇 가지 설정 작업을 완료합니다.
  - 라이센스를 설정합니다.
  - 첫 번째 클러스터를 추가합니다.
  - 클러스터를 추가할 때 검색된 스토리지 백엔드를 추가합니다.
  - 앱 백업을 저장할 오브젝트 저장소 버킷을 추가합니다.

에 대해 자세히 알아보십시오 "Astra Control Center를 설정합니다".

앱을 클러스터에 추가할 수 있습니다. 클러스터에 이미 관리 중인 앱이 있으면 Astra Control Center를 사용하여 관리할 수 있습니다. 그런 다음 Astra Control Center를 사용하여 스냅샷, 백업, 클론 및 복제 관계를 생성합니다.

를 참조하십시오

- "Astra Control Service 문서"
- "Astra Control Center 문서"
- "Astra Trident 문서"
- "Astra Control API 설명서"
- "ONTAP 설명서"

Astra Control Center 요구 사항

먼저 운영 환경, 애플리케이션 클러스터, 애플리케이션, 라이센스 및 웹 브라우저의 준비 상태를 확인하십시오. Astra Control Center를 구축하고 운영하는 데 필요한 요구 사항을 사용자 환경이 충족하는지 확인합니다.

지원되는 호스트 클러스터 **Kubernetes** 환경

Astra Control Center는 다음과 같은 Kubernetes 호스트 환경에서 검증되었습니다.

> Astra Control Center를 호스팅하도록 선택한 Kubernetes 환경이 환경의 공식 문서에 설명된 기본 리소스 요구사항을 충족하는지 확인합니다.

호스트 클러스터에서의 **Kubernetes** 배포	지원되는 버전
Azure Stack HCI 기반 Azure Kubernetes Service	AKS 1.24.11 ~ 1.26.6이 포함된 Azure Stack HCI 21H2 및 22H2
Google Anthos	1.15 ~ 1.16(참조 Google Anthos 수신 요구 사항)
Kubernetes(업스트림)	1.27 ~ 1.29

호스트 클러스터에서의 **Kubernetes** 배포	지원되는 버전
RKE(Rancher Kubernetes Engine)	RKE 1: Rancher Manager 2.7.9 포함 버전 1.24.17, 1.25.13, 1.26.8 RKE 2: Rancher Manager 2.6.13이 있는 버전 1.23.16 및 1.24.13 RKE 2: Rancher Manager 2.7.9 포함 버전 1.24.17, 1.25.14, 1.26.9
Red Hat OpenShift Container Platform	4.12에서 4.14까지

호스트 클러스터 리소스 요구 사항

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

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

- * CPU 확장 *: 호스팅 환경의 모든 노드에 있는 CPU에는 AVX 확장이 활성화되어 있어야 합니다.
- * 작업자 노드 *: 총 3개 이상의 작업자 노드, CPU 코어 4개, 12GB RAM
- * 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.
  - 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"
  - 를 사용합니다 `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.
  - 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.
  - TKG 또는 TKGi 환경에 Astra Control Center를 구축할 때 Astra Control Provisioner의 특정 요구사항을 숙지하십시오.
    - 클러스터는 권한이 있는 워크로드를 지원해야 합니다.
    - 를 클릭합니다 `--kubelet-dir` 플래그를 kubelet 디렉터리의 위치로 설정해야 합니다. 기본적으로 이 값은 입니다 `/var/vcap/data/kubelet`.
    - 를 사용하여 kubelet 위치 지정 `--kubelet-dir` Trident Operator, Helm 및 에 대해 작업하는 것으로 알려져 있습니다 `tridentctl` 적합합니다.

서비스 메시 요구 사항

Astra Control Center 호스트 클러스터에 지원되는 Istio 서비스 메시의 바닐라 버전을 설치하는 것이 좋습니다. 을 참조하십시오 "지원되는 릴리스" 지원되는 Istio 버전 OpenShift Service Mesh와 같은 자사 서비스 메시의 브랜드 릴리스는 Astra Control Center에서 검증되지 않았습니다.

Astra Control Center를 호스트 클러스터에 설치된 Istio 서비스 메시와 통합하려면 Astra Control Center의 일부로 통합해야 합니다 "설치" 그리고 이 과정에 독립적이지 않습니다.

> 호스트 클러스터에 서비스 메시를 구성하지 않고 Astra Control Center를 설치하여 사용하는 경우 심각한 보안 문제가 발생할 수 있습니다.

아스트라 트리덴트

이 릴리즈에서 Astra Control Provisioner 대신 Astra Trident를 사용하려면 Astra Trident 23.04 이상 버전이 지원됩니다. Astra Control Center가 필요합니다 Astra Control Provisioner 향후 릴리스에서.

Astra Control Provisioner

Astra Control Provisioner 고급 스토리지 기능을 사용하려면 Astra Trident 23.10 이상을 설치하고 를 사용하도록 설정해야 합니다 "Astra Control Provisioner 기능". 최신 Astra Control Provisioner 기능을 사용하려면 Astra Trident 및 Astra Control Center의 최신 버전이 필요합니다.

- * Astra Control Center와 함께 사용할 수 있는 최소 Astra Control Provisioner 버전 *: Astra Control Provisioner 23.10 이상이 설치 및 구성되었습니다.

Astra Trident의 ONTAP 구성

- * 스토리지 클래스 *: 클러스터에 하나 이상의 스토리지 클래스를 구성합니다. 기본 스토리지 클래스가 구성된 경우 기본 지정의 유일한 스토리지 클래스인지 확인합니다.
- * 스토리지 드라이버 및 작업자 노드 *: 포드가 백엔드 스토리지와 상호 작용할 수 있도록 클러스터의 작업자 노드를 적절한 스토리지 드라이버로 구성해야 합니다. Astra Control Center는 Astra Trident에서 제공하는 다음과 같은 ONTAP 드라이버를 지원합니다.
  - `ontap-nas`
  - `ontap-san`
  - `ontap-san-economy` (이 스토리지 클래스 유형에서는 애플리케이션 복제를 사용할 수 없습니다.)
  - `ontap-nas-economy` (이 스토리지 클래스 유형에서는 스냅샷 및 애플리케이션 복제 정책을 사용할 수 없습니다.)

스토리지 백엔드

지원되는 백엔드에 충분한 용량이 있는지 확인합니다.

- * 필요한 스토리지 백엔드 용량 *: 500GB 이상 사용 가능
- * 지원되는 백엔드 *: Astra Control Center는 다음과 같은 스토리지 백엔드를 지원합니다.
  - NetApp ONTAP 9.9.1 이상 AFF, FAS 및 ASA 시스템
  - NetApp ONTAP Select 9.9.1 이상
  - NetApp Cloud Volumes ONTAP 9.9.1 이상
  - (Astra Control Center 기술 미리보기용) 데이터 보호 작업을 위한 NetApp ONTAP 9.10.1 이상
  - Longhorn 1.5.0 이상
    - VolumeSnapshotClass 객체를 수동으로 생성해야 합니다. 을 참조하십시오 "롱혼 설명서" 를

참조하십시오.

- NetApp MetroCluster
  - 관리 Kubernetes 클러스터는 확장 구성에 있어야 합니다.
- 스토리지 백엔드는 지원되는 클라우드 공급자를 통해 제공됩니다

ONTAP 라이센스

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 확인하려면 을 참조하십시오 "ONTAP 라이센스 관리".

NetApp MetroCluster

NetApp MetroCluster을 스토리지 백엔드로 사용하는 경우 다음을 수행해야 합니다.

- 사용하는 Astra Trident 드라이버에서 SVM 관리 LIF를 백엔드 옵션으로 지정합니다
- 적절한 ONTAP 라이센스가 있는지 확인합니다

MetroCluster LIF를 구성하려면 각 드라이버에 대한 다음 옵션과 예를 참조하십시오.

- "산"
- "NAS"

Astra Control Center 라이센스

Astra Control Center에는 Astra Control Center 라이센스가 필요합니다. Astra Control Center를 설치할 때 4,800 CPU 장치에 대한 90일 평가 라이센스가 내장되어 있습니다. 용량 또는 다른 평가 조건이 필요하거나 전체 라이센스로 업그레이드하려는 경우 NetApp에서 다른 평가 라이센스 또는 전체 라이센스를 얻을 수 있습니다. 애플리케이션과 데이터를 보호하려면 라이센스가 필요합니다.

Astra Control Center는 무료 평가판을 신청하여 사용해 볼 수 있습니다. 등록을 통해 등록할 수 있습니다 "여기".

라이센스를 설정하려면 을 참조하십시오 "90일 평가판 라이센스를 사용합니다".

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

네트워킹 요구 사항

Astra Control Center가 올바르게 통신할 수 있도록 운영 환경을 구성합니다. 다음 네트워킹 구성이 필요합니다.

- * FQDN 주소 *: Astra Control Center에 대한 FQDN 주소가 있어야 합니다.
- * 인터넷 액세스 *: 인터넷에 대한 외부 액세스 권한이 있는지 여부를 확인해야 합니다. 그렇지 않으면 에 지원 번들을 보내는 등 일부 기능이 제한될 수 있습니다 "NetApp Support 사이트".

- * 포트 액세스 *: 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 Center와 Astra Control Center에 액세스하는 데 사용되는 시스템 간에 이 포트가 양방향으로 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	Amazon S3 스토리지 버킷 공급자	443	HTTPS	Amazon S3 스토리지 통신
Astra 제어 센터	NetApp AutoSupport를 참조하십시오	443	HTTPS	NetApp AutoSupport 커뮤니케이션
Astra 제어 센터	관리형 Kubernetes 클러스터	443/6443 을 참조하십시오 * 참고 *: 관리되는 클러스터에서 사용하는 포트는 클러스터에 따라 다를 수 있습니다. 클러스터 소프트웨어 공급업체의 설명서를 참조하십시오.	HTTPS	관리형 클러스터와의 통신 - Astra Control Center를 호스팅하는 클러스터와 관리되는 각 클러스터 간에 이 포트가 두 방식으로 열려 있는지 확인합니다

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.

로드 밸런서는 Astra Control Center 작업자 노드 IP 주소와 동일한 서브넷에 있는 IP 주소를 사용해야 합니다.

자세한 내용은 을 참조하십시오 "부하 분산을 위한 수신 설정".

Google Anthos 수신 요구 사항

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

지원되는 웹 브라우저

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

애플리케이션 클러스터에 대한 추가 요구사항

Astra Control Center 기능을 사용하려는 경우 다음 요구 사항을 염두에 두십시오.

- * 애플리케이션 클러스터 요구 사항 *: "클러스터 관리 요구 사항"
  - * 관리되는 애플리케이션 요구 사항 *: "설명합니다"
  - * 애플리케이션 복제에 대한 추가 요구 사항 *: "복제 사전 요구 사항"

다음 단계

를 봅니다 "빠른 시작" 개요.

Astra Control Center를 빠르게 시작합니다

Astra Control Center를 시작하는 데 필요한 단계를 간략하게 소개합니다. 각 단계의 링크를 클릭하면 자세한 내용을 제공하는 페이지로 이동합니다.

1 **Kubernetes** 클러스터 요구사항을 검토하십시오

귀사의 환경이 다음 요구 사항을 충족하는지 확인하십시오.

- Kubernetes 클러스터 *
- "호스트 클러스터가 운영 환경 요구 사항을 충족하는지 확인합니다"
- "온프레미스 Kubernetes 클러스터의 로드 밸런싱을 위해 수신 구성"
- 스토리지 통합 *
- "환경에 Astra Control Provisioner가 포함되어 있는지 확인합니다"
- "Astra Control Provisioner 고급 관리 및 스토리지 프로비저닝 기능을 지원합니다"
- "클러스터 작업자 노드를 준비합니다"
- "스토리지 백엔드를 구성합니다"

- "스토리지 클래스를 구성합니다"
- "볼륨 스냅샷 컨트롤러를 설치합니다"
- "볼륨 스냅샷 클래스를 생성합니다"
- ONTAP 자격 증명 *
- "ONTAP 자격 증명을 구성합니다"

2 **Astra Control Center**를 다운로드하여 설치합니다

다음 설치 작업을 완료합니다.

- "NetApp Support 사이트 다운로드 페이지에서 Astra Control Center를 다운로드합니다"
- NetApp 라이센스 파일을 얻습니다.
  - Astra Control Center를 평가하는 경우 이미 포함된 평가 라이센스가 포함되어 있습니다
  - "이미 Astra Control Center를 구입한 경우 라이센스 파일을 생성합니다"
- "Astra Control Center를 설치합니다"
- "추가 옵션 구성 단계를 수행합니다"

3 몇 가지 초기 설정 작업을 완료합니다

시작하려면 몇 가지 기본 작업을 완료하십시오.

- "라이센스를 추가합니다"
- "클러스터 관리를 위한 환경을 준비합니다"
- "클러스터를 추가합니다"
- "스토리지 백엔드를 추가합니다"
- "버킷을 추가합니다"

4 **Astra Control Center**를 사용합니다

Astra Control Center 설정을 마친 후 Astra Control UI 또는 를 사용합니다 "Astra Control API를 참조하십시오" 앱 관리 및 보호를 시작하려면

- "계정 관리"사용자, 역할, LDAP, 자격 증명 등
- "알림을 관리합니다"
- "앱 관리": 관리할 리소스를 정의합니다.
- "앱 보호"보호 정책을 구성하고 앱을 복제, 클론 복제 및 마이그레이션합니다.

를 참조하십시오

- "Astra Control API를 사용합니다"

- "Astra Control Center를 업그레이드합니다"
- "Astra Control에 대한 도움을 받으십시오"

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"

환경에 따라 Astra Control Center를 설치한 후 추가 구성이 필요할 수 있습니다.

- "설치 후 Astra Control Center를 구성합니다"

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

Astra Control Center를 설치하려면 설치 이미지를 다운로드하고 다음 단계를 수행하십시오. 이 절차를 사용하여 인터넷에 연결되었거나 공기가 연결된 환경에 Astra Control Center를 설치할 수 있습니다.

Astra Control Center 설치 프로세스의 데모는 를 참조하십시오 "이 비디오".

시작하기 전에

- * 환경 필수 조건 충족 *: "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".

Astra Control Center를 세 번째 고장 도메인 또는 보조 사이트에 배포합니다. 앱 복제 및 원활한 재해 복구에 권장됩니다.

- * 건강한 서비스 보장 * : 모든 API 서비스가 정상 상태이고 사용 가능한지 확인하십시오.

```
kubectl get apiservices
```

- * 라우팅 가능한 FQDN *: 사용하려는 Astra FQDN을 클러스터로 라우팅할 수 있습니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.
- * 인증서 관리자 구성 *: 클러스터에 이미 인증서 관리자가 있는 경우 일부 작업을 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하려고 시도하지 않습니다. 기본적으로 Astra Control Center는 설치 중에 자체 인증서 관리자를 설치합니다.
- * (ONTAP SAN 드라이버만 해당) 다중 경로 사용 *: ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

다음 사항도 고려해야 합니다.

- * NetApp Astra Control 이미지 레지스트리에 액세스 *:

NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

   계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

c. Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- * 보안 통신을 위한 서비스 메시를 설치합니다 * : Astra Control 호스트 클러스터 통신 채널은 을 사용하여 보안을 유지하는 것이 좋습니다 "지원되는 서비스 메시입니다".

> Astra Control Center를 서비스 메시와 통합하는 작업은 Astra Control Center 중에만 수행할 수 있습니다 "설치" 그리고 이 과정에 독립적이지 않습니다. 메시에서 메시되지 않은 환경으로 다시 변경하는 것은 지원되지 않습니다.

Istio 서비스 메시를 사용하려면 다음을 수행해야 합니다.

  - 를 추가합니다 `istio-injection:enabled` 라벨 Astra Control Center를 구축하기 전에 Astra 네임스페이스에 매핑
  - 를 사용합니다 `Generic` 수신 설정 에 대한 대체 침입을 제공합니다 외부 부하 균형.
  - Red Hat OpenShift 클러스터의 경우 을 정의해야 합니다 `NetworkAttachmentDefinition` 연결된 모든 Astra Control Center 네임스페이스에서 (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` 응용 프로그램 클러스터 또는 대체된 사용자 지정 네임스페이스의 경우).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center를 다운로드하고 압축을 풉니다
- 로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다
- 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 운영자를 삭제하지 마십시오(예: `kubectl delete -f astra_control_center_operator_deploy.yaml`) 포드가 삭제되지 않도록 Astra Control Center 설치 또는 작동 중에 언제든지.

Astra Control Center를 다운로드하고 압축을 풉니다

다음 위치 중 하나에서 Astra Control Center 이미지를 다운로드하십시오.

- * Astra 컨트롤 서비스 이미지 레지스트리 *: Astra 컨트롤 센터 이미지에 로컬 레지스트리를 사용하지 않거나 NetApp Support 사이트에서 번들 다운로드보다 이 방법을 선호하는 경우 이 옵션을 사용합니다.
- * NetApp Support 사이트 *: Astra 컨트롤 센터 이미지와 함께 로컬 레지스트리를 사용하는 경우 이 옵션을 사용합니다.

Astra Control 이미지 레지스트리

1. Astra Control Service에 로그인합니다.
2. 대시보드에서 * Astra Control의 자가 관리형 인스턴스 배포 * 를 선택합니다.
3. 지침에 따라 Astra Control 이미지 레지스트리에 로그인하고 Astra Control Center 설치 이미지를 가져온 다음 이미지를 추출합니다.

NetApp Support 사이트

1. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장되지만 선택 사항) Astra Control Center용 인증서 및 서명 번들을 다운로드합니다 (`astra-control-center-certs-[version].tar.gz`)를 클릭하여 번들 서명을 확인합니다.

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

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

   출력이 표시됩니다 `Verified OK` 확인 성공 후.

3. Astra Control Center 번들에서 이미지를 추출합니다.

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

로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다

Astra Control Center 번들을 로컬 레지스트리에 푸시하려는 경우 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

NetApp Astra kubtl 플러그인을 설치합니다

최신 NetApp Astra kubectl 명령줄 플러그인을 설치하려면 다음 단계를 완료하십시오.

시작하기 전에

NetApp은 다양한 CPU 아키텍처 및 운영 체제에 대한 플러그인 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다.

이전 설치에서 이미 플러그인을 설치한 경우 "최신 버전이 있는지 확인하십시오" 다음 단계를 수행하기 전에

단계

1. 사용 가능한 NetApp Astra kubeck 플러그인 바이너리를 나열합니다.

   kubbeck 플러그인 라이브러리는 tar 번들의 일부이며 폴더에 압축이 풀립니다 `kubectl-astra`.

   ```
   ls kubectl-astra/
   ```

2. 운영 체제 및 CPU 아키텍처에 필요한 파일을 현재 경로로 이동하고 이름을 로 변경합니다 `kubectl-astra`:

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

레지스트리에 이미지를 추가합니다

1. Astra Control Center 번들을 로컬 레지스트리로 푸시하려는 경우 컨테이너 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

a. 타볼의 루트 디렉토리로 변경합니다. 가 표시됩니다 `acc.manifest.bundle.yaml` 파일 및 다음 디렉토리:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. Astra Control Center 이미지 디렉토리의 패키지 이미지를 로컬 레지스트리에 밀어 넣습니다. 를 실행하기 전에 다음 대체 작업을 수행합니다 `push-images` 명령:

- <BUNDLE_FILE>를 Astra Control 번들 파일의 이름으로 바꿉니다 (`acc.manifest.bundle.yaml`)를 클릭합니다.
- <MY_FULL_REGISTRY_PATH>를 Docker 저장소의 URL로 바꿉니다. 예를 들어, "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>".
- <MY_REGISTRY_USER>를 사용자 이름으로 바꿉니다.
- <MY_REGISTRY_TOKEN>를 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

팟맨

a. 타볼의 루트 디렉토리로 변경합니다. 이 파일과 디렉토리가 표시됩니다.

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 레지스트리에 로그인합니다.

```
podman login <YOUR_REGISTRY>
```

c. 사용하는 Podman 버전에 맞게 사용자 지정된 다음 스크립트 중 하나를 준비하고 실행합니다. <MY_FULL_REGISTRY_PATH>를 모든 하위 디렉토리가 포함된 리포지토리의 URL로 대체합니다.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

ⓘ 레지스트리 구성에 따라 스크립트가 만드는 이미지 경로는 다음과 같아야 합니다.

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 디렉토리를 변경합니다.

```
cd manifests
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. Astra Control Center 호스트 클러스터의 kubecononfig 내보내기:

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

설치를 완료하기 전에 Astra Control Center를 설치할 클러스터를 추천하십시오.

2. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

   a. 'NetApp-acc-operator' 네임스페이스 생성:

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

   b. NetApp-acc-operator 네임스페이스에 대한 암호를 생성합니다. Docker 정보를 추가하고 다음 명령을 실행합니다.

자리 표시자입니다 `your_registry_path` 이전에 업로드한 이미지의 위치와 일치해야 합니다(예: `[Registry_URL]/netapp/astra/astracc/24.02.0-69`)를 클릭합니다.

```
kubectl create secret docker-registry astra-registry-cred -n netapp-acc-operator --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

+

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

+

암호를 생성한 후 네임스페이스를 삭제하면 네임스페이스를 다시 만든 다음 네임스페이스에 대한 암호를 다시 생성합니다.

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

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

   b. 에 대한 암호를 만듭니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다. Docker 정보를 추가하고 레지스트리 기본 설정에 따라 적절한 명령 중 하나를 실행합니다.

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

```
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]
```

Astra Control Center 운영자를 설치합니다

1. (로컬 레지스트리만 해당) 로컬 레지스트리를 사용하는 경우 다음 단계를 수행하십시오.

   a. Astra Control Center 운영자 배포 YAML을 엽니다.

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   ⓘ YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

   b. 인증이 필요한 레지스트리를 사용하는 경우 'imagePullSecrets:[]'의 기본 줄을 다음과 같이 바꿉니다.

   ```
   imagePullSecrets: [{name: astra-registry-cred}]
   ```

   c. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `kube-rbac-proxy` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

   d. 변경 `ASTRA_IMAGE_REGISTRY` 의 경우 `acc-operator-controller-manager` 이미지를 에서 푸시한 레지스트리 경로로 이미지 이전 단계.

```
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
  strategy:
    type: Recreate
```

```
  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: ASTRA_IMAGE_REGISTRY/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
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
```

```
            memory: 75Mi
          securityContext:
            allowPrivilegeEscalation: false
        imagePullSecrets: []
        securityContext:
          runAsUser: 65532
        terminationGracePeriodSeconds: 10
```

2. Astra Control Center 운영자를 설치합니다.

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

샘플 응답을 위해 확장:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.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
```

3. Pod가 실행 중인지 확인합니다.

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

Astra Control Center를 구성합니다

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

```
vim astra_control_center.yaml
```

YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

2. 다음 설정을 수정하거나 확인합니다.

계정 이름

설정	지침	유형	예
`accountName`	를 변경합니다 `accountName` Astra Control Center 계정과 연결할 이름에 대한 문자열입니다. 하나의 accountName만 있을 수 있습니다.	문자열	`Example`

astraVersion을 참조하십시오

설정	지침	유형	예
`astraVersion`	배포할 Astra Control Center의 버전입니다. 값이 미리 채워질 수 있으므로 이 설정에 대한 작업은 필요하지 않습니다.	문자열	`24.02.0-69`

astraAddress를 선택합니다

설정	지침	유형	예
`astraAddress`	를 변경합니다 `astraAddress` 브라우저에서 Astra Control Center에 액세스하기 위해 사용할 FQDN(권장) 또는 IP 주소에 대한 문자열입니다. 이 주소는 Astra Control Center가 데이터 센터에서 어떻게 검색되는지 정의하며, 이 주소를 완료하면 로드 밸런서에서 제공한 것과 동일한 FQDN 또는 IP 주소입니다 "Astra Control Center 요구 사항". 참고: 사용하지 마십시오 `http://` 또는 `https://` 를 입력합니다. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.	문자열	`astra.example.com`

AutoSupport

이 섹션에서 선택한 항목에 따라 NetApp의 사전 지원 응용 프로그램인 디지털 어드바이저에 참여할지 여부와 데이터 전송 위치가 결정됩니다. 인터넷 연결이 필요하며(포트 442) 모든 지원 데이터가 익명화됩니다.

설정	사용	지침	유형	예
`autoSupport.enrolled`	둘 다 가능합니다 `enrolled` 또는 `url` 필드를 선택해야 합니다	변경 `enrolled` 을 눌러 AutoSupport to로 이동합니다 `false` 인터넷 연결이 없거나 보관되지 않은 사이트의 경우 `true` 연결된 사이트의 경우. 의 설정 `true` 지원을 위해 익명 데이터를 NetApp에 전송할 수 있습니다. 기본 선택 옵션은 입니다 `false` 및 은 NetApp에 지원 데이터가 전송되지 않음을 나타냅니다.	부울	`false` (이 값은 기본값입니다.)
`autoSupport.url`	둘 다 가능합니다 `enrolled` 또는 `url` 필드를 선택해야 합니다	이 URL은 익명 데이터를 보낼 위치를 결정합니다.	문자열	`https://support.netapp.com/asupprod/post/1.0/postAsup`

이메일

설정	지침	유형	예
`email`	를 변경합니다 `email` 문자열을 기본 초기 관리자 주소로 설정합니다. 에서 사용할 이 이메일 주소를 복사합니다 나중에. 이 이메일 주소는 UI에 로그인할 초기 계정의 사용자 이름으로 사용되며 Astra Control에서 이벤트를 알립니다.	문자열	`admin@example.com`

이름

설정	지침	유형	예
`firstName`	Astra 계정과 연결된 기본 초기 관리자의 이름입니다. 여기에 사용된 이름은 처음 로그인한 후 UI의 제목에 표시됩니다.	문자열	`SRE`

성

설정	지침	유형	예
`lastName`	Astra 계정과 연결된 기본 초기 관리자의 성. 여기에 사용된 이름은 처음 로그인한 후 UI의 제목에 표시됩니다.	문자열	`Admin`

imageRegistry(이미지 레지스트리**)**

이 섹션에서 선택한 사항은 Astra 응용 프로그램 이미지, Astra Control Center Operator 및 Astra Control Center Helm 리포지토리를 호스팅하는 컨테이너 이미지 레지스트리를 정의합니다.

설정	사용	지침	유형	예
`imageRegistry.name`	필수 요소입니다	Astra Control Center 배포에 필요한 모든 이미지를 호스팅하는 Astra Control 이미지 레지스트리의 이름입니다. 이 값은 미리 채워지며 로컬 레지스트리를 구성하지 않으면 아무 작업도 필요하지 않습니다. 로컬 레지스트리의 경우 이 기존 값을 에서 이미지를 푸시한 이미지 레지스트리 이름으로 바꿉니다 이전 단계. 사용하지 마십시오 `http://` 또는 `https://` 레지스트리 이름.	문자열	`cr.astra.netapp.io` (기본값) `example.registry.com/astra` (로컬 레지스트리 예)

설정	사용	지침	유형	예
`imageRegistry.secret`	선택 사항	이미지 레지스트리를 인증하는 데 사용되는 Kubernetes 비밀의 이름입니다. 값은 미리 채워지며 로컬 레지스트리와 에서 해당 레지스트리에 대해 입력한 문자열을 구성하지 않으면 아무 작업도 필요하지 않습니다 `imageRegistry.name` 비밀이 필요합니다. 중요: 인증이 필요하지 않은 로컬 레지스트리를 사용하는 경우 이를 삭제해야 합니다 `secret` 줄 내부 `imageRegistry` 그렇지 않으면 설치가 실패합니다.	문자열	`astra-registry-cred`

storageClass 를 선택합니다

설정	지침	유형	예
`storageClass`	를 변경합니다 `storageClass` 값 시작 `ontap-gold` 설치 시 필요한 다른 storageClass 리소스로 이동합니다. 명령을 실행합니다 `kubectl get sc` 구성된 기존 스토리지 클래스를 확인하려면 다음을 수행합니다. Astra Control Provisioner로 구성된 스토리지 클래스 중 하나를 매니페스트 파일에 입력해야 합니다 (`astra-control-center-<version>.manifest`) 및 는 Astra PVS에 사용됩니다. 이 옵션이 설정되어 있지 않으면 기본 스토리지 클래스가 사용됩니다. 참고: 기본 스토리지 클래스가 구성된 경우 기본 주석이 있는 유일한 스토리지 클래스인지 확인하십시오.	문자열	`ontap-gold`

볼륨 리클레이임정책

설정	지침	유형	옵션
`volumeReclaimPolicy`	그러면 Astra의 PVS에 대한 재확보 정책이 설정됩니다. 이 정책을 으로 설정합니다 `Retain` Astra가 삭제된 후 영구 볼륨을 유지합니다. 이 정책을 으로 설정합니다 `Delete` Astra가 삭제된 후 영구 볼륨을 삭제합니다. 이 값을 설정하지 않으면 PVS가 유지됩니다.	문자열	• `Retain` (기본값) • `Delete`

응력 유형

설정	지침	유형	옵션
`ingressType`	다음 수신 유형 중 하나를 사용하십시오. * 일반 * (`ingressType: "Generic"`) (기본값) 다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 구축한 후 를 구성해야 합니다 "수신 컨트롤러" URL을 사용하여 Astra Control Center를 표시합니다. 중요: Astra Control Center에서 서비스 메시를 사용하려면 을 선택해야 합니다 `Generic` 수신 유형 으로 설정하고 직접 설정합니다 "수신 컨트롤러". * AccTraefik * (`ingressType: "AccTraefik"`) 수신 컨트롤러를 구성하지 않으려는 경우 이 옵션을 사용하십시오. 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	문자열	• `Generic` (기본값) • `AccTraefik`

스케일크기

설정	지침	유형	옵션
scaleSize	기본적으로 Astra는 HA(High Availability)를 사용합니다. scaleSize의 Medium`즉, HA에서 대부분의 서비스를 구축하고 이중화를 위해 여러 복제본을 배포합니다. 와 함께 `scaleSize 현재 Small`Astra는 소비를 줄이기 위한 필수 서비스를 제외한 모든 서비스의 복제본 수를 줄일 것입니다. 팁: `Medium 약 100개의 Pod로 구축 가능(임시 워크로드 제외) 100 Pod는 3개의 마스터 노드 및 3개의 작업자 노드 구성을 기반으로 합니다.) 특히 재해 복구 시나리오를 고려할 때 사용자 환경에서 문제가 될 수 있는 Pod별 네트워크 제한 사항에 유의하십시오.	문자열	• Small • Medium (기본값)

astraResourcaresaTM

설정	지침	유형	옵션
`astraResourcesScaler`	AstraControlCenter 리소스 제한에 대한 확장 옵션 기본적으로 Astra Control Center는 Astra 내의 대부분의 구성 요소에 대해 설정된 리소스 요청과 함께 배포됩니다. 이 구성을 통해 Astra Control Center 소프트웨어 스택은 애플리케이션 로드 및 확장 수준이 높은 환경에서 더 나은 성능을 발휘할 수 있습니다. 그러나 더 작은 개발 또는 테스트 클러스터를 사용하는 시나리오에서는 CR 필드를 사용합니다 `astraResourcesScalar` 로 설정할 수 있습니다 `Off`. 이렇게 하면 리소스 요청이 비활성화되고 소규모 클러스터에 구축할 수 있습니다.	문자열	• `Default` (기본값) • `Off`

추가 가치입니다

설치 시 알려진 문제를 방지하려면 Astra Control Center CR에 다음 추가 값을 추가합니다.

```
additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

CRD

이 섹션에서 선택한 사항은 Astra Control Center에서 CRD를 처리하는 방법을 결정합니다.

설정	지침	유형	예
`crds.externalCertManager`	외부 인증서 관리자를 사용하는 경우 를 변경합니다 `externalCertManager` 를 선택합니다 `true`. 기본값입니다 `false` 설치 중에 Astra Control Center가 자체 인증서 관리자 CRD를 설치합니다. CRD는 클러스터 전체 오브젝트이며 이를 설치하면 클러스터의 다른 부분에 영향을 줄 수 있습니다. 이 플래그를 사용하여 Astra Control Center에 이러한 CRD가 Astra Control Center 외부의 클러스터 관리자에 의해 설치 및 관리된다는 신호를 보낼 수 있습니다.	부울	`False` (이 값은 기본값입니다.)
`crds.externalTraefik`	기본적으로 Astra Control Center는 필요한 Traefik CRD를 설치합니다. CRD는 클러스터 전체 오브젝트이며 이를 설치하면 클러스터의 다른 부분에 영향을 줄 수 있습니다. 이 플래그를 사용하여 Astra Control Center에 이러한 CRD가 Astra Control Center 외부의 클러스터 관리자에 의해 설치 및 관리된다는 신호를 보낼 수 있습니다.	부울	`False` (이 값은 기본값입니다.)

설치를 완료하기 전에 구성에 맞는 올바른 스토리지 클래스 및 수신 유형을 선택했는지 확인하십시오.

Astra_CONTROL_CENTER.YAML을 샘플링합니다

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  crds:
    externalTraefik: false
    externalCertManager: false
```

Astra 제어 센터 및 운전자 설치를 완료합니다

1. 이전 단계에서 작성하지 않은 경우, "NetApp-acc"(또는 사용자 지정) 네임스페이스를 작성하십시오.

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

2. Astra Control Center에서 서비스 메시를 사용하는 경우 에 다음 레이블을 추가합니다 `netapp-acc` 또는 사용자 지정 네임스페이스:

   수신 유형입니다 (`ingressType`)를 로 설정해야 합니다 `Generic` Astra Control Center CR에서 이 명령을 진행하기 전에

```
kubectl label ns [netapp-acc or custom namespace] istio-
injection:enabled
```

3. (권장) "엄격한 MTL을 활성화합니다" Istio 서비스 메시의 경우:

```
kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
spec:
  mtls:
    mode: STRICT
EOF
```

4. "NetApp-acc"(또는 사용자 지정) 네임스페이스에 Astra Control Center를 설치합니다.

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

Astra Control Center 운영자는 환경 요구 사항에 대한 자동 검사를 실행합니다. 없습니다 "요구 사항" 설치가 실패하거나 Astra Control Center가 제대로 작동하지 않을 수 있습니다. 를 참조하십시오 다음 섹션을 참조하십시오 자동 시스템 점검과 관련된 경고 메시지를 확인합니다.

시스템 상태를 확인합니다

kubeck 명령을 사용하여 시스템 상태를 확인할 수 있습니다. OpenShift를 사용하려는 경우 검증 단계에 유사한 OC 명령을 사용할 수 있습니다.

단계

1. 설치 프로세스에서 유효성 검사와 관련된 경고 메시지가 생성되지 않았는지 확인합니다.

```
kubectl get acc [astra or custom Astra Control Center CR name] -n
[netapp-acc or custom namespace] -o yaml
```

Astra Control Center 운영자 로그에도 추가 경고 메시지가 표시됩니다.

2. 자동화된 요구 사항 확인을 통해 보고된 환경 관련 문제를 모두 해결하십시오.

사용자 환경이 을(를) 충족하는지 확인하여 문제를 해결할 수 있습니다 "요구 사항" Astra Control Center의 경우

3. 모든 시스템 구성 요소가 성공적으로 설치되었는지 확인합니다.

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

각 포드는 'Running' 상태여야 합니다. 시스템 포드를 구축하는 데 몇 분 정도 걸릴 수 있습니다.

샘플 응답을 위해 확장합니다

```
acc-helm-repo-5bd77c9ddd-8wxm2              1/1     Running     0
1h
activity-5bb474dc67-8l9ss                   1/1     Running     0
1h
activity-5bb474dc67-qbrtq                   1/1     Running     0
1h
api-token-authentication-6wbj2              1/1     Running     0
1h
api-token-authentication-9pgw6              1/1     Running     0
1h
api-token-authentication-tqf6d              1/1     Running     0
1h
asup-5495f44dbd-z4kft                       1/1     Running     0
1h
authentication-6fdd899858-5x45s             1/1     Running     0
1h
bucketservice-84d47487d-n9xgp               1/1     Running     0
1h
bucketservice-84d47487d-t5jhm               1/1     Running     0
1h
cert-manager-5dcb7648c4-hbldc               1/1     Running     0
1h
cert-manager-5dcb7648c4-nr9qf               1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-bk2tf    1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-pfnck    1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-ngz2x        1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-rwtbn        1/1     Running     0
1h
certificates-5f5b7b4dd-52tnj                1/1     Running     0
1h
certificates-5f5b7b4dd-gtjbx                1/1     Running     0
1h
certificates-expiry-check-28477260-dz5vw    0/1     Completed   0
1h
cloud-extension-6f58cc579c-lzfmv            1/1     Running     0
1h
cloud-extension-6f58cc579c-zw2km            1/1     Running     0
1h
cluster-orchestrator-79dd5c8d95-qjg92       1/1     Running     0
```

```
1h
composite-compute-85dc84579c-nz82f          1/1     Running     0
1h
composite-compute-85dc84579c-wx2z2          1/1     Running     0
1h
composite-volume-bff6f4f76-789nj            1/1     Running     0
1h
composite-volume-bff6f4f76-kwnd4            1/1     Running     0
1h
credentials-79fd64f788-m7m8f                1/1     Running     0
1h
credentials-79fd64f788-qnc6c                1/1     Running     0
1h
entitlement-f69cdbd77-4p2kn                 1/1     Running     0
1h
entitlement-f69cdbd77-hswm6                 1/1     Running     0
1h
features-7b9585444c-7xd7m                   1/1     Running     0
1h
features-7b9585444c-dcqwc                   1/1     Running     0
1h
fluent-bit-ds-crq8m                         1/1     Running     0
1h
fluent-bit-ds-gmgq8                         1/1     Running     0
1h
fluent-bit-ds-gzr4f                         1/1     Running     0
1h
fluent-bit-ds-j6sf6                         1/1     Running     0
1h
fluent-bit-ds-v4t9f                         1/1     Running     0
1h
fluent-bit-ds-x7j59                         1/1     Running     0
1h
graphql-server-6cc684fb46-2x8lr             1/1     Running     0
1h
graphql-server-6cc684fb46-bshbd             1/1     Running     0
1h
hybridauth-84599f79fd-fjc7k                 1/1     Running     0
1h
hybridauth-84599f79fd-s9pmn                 1/1     Running     0
1h
identity-95df98cb5-dvlmz                    1/1     Running     0
1h
identity-95df98cb5-krf59                    1/1     Running     0
1h
influxdb2-0                                 1/1     Running     0
```

```
1h
keycloak-operator-6d4d688697-cfq8b          1/1     Running     0
1h
krakend-5d5c8f4668-7bq8g                    1/1     Running     0
1h
krakend-5d5c8f4668-t8hbn                    1/1     Running     0
1h
license-689cdd4595-2gsc8                    1/1     Running     0
1h
license-689cdd4595-g6vwk                    1/1     Running     0
1h
login-ui-57bb599956-4fwgz                   1/1     Running     0
1h
login-ui-57bb599956-rhztb                   1/1     Running     0
1h
loki-0                                      1/1     Running     0
1h
metrics-facade-846999bdd4-f7jdm             1/1     Running     0
1h
metrics-facade-846999bdd4-lnsxl             1/1     Running     0
1h
monitoring-operator-6c9d6c4b8c-ggkrl        2/2     Running     0
1h
nats-0                                      1/1     Running     0
1h
nats-1                                      1/1     Running     0
1h
nats-2                                      1/1     Running     0
1h
natssync-server-6df7d6cc68-9v2gd            1/1     Running     0
1h
nautilus-64b7fbdd98-bsgwb                   1/1     Running     0
1h
nautilus-64b7fbdd98-djlhw                   1/1     Running     0
1h
openapi-864584bccc-75nlv                    1/1     Running     0
1h
openapi-864584bccc-zh6bx                    1/1     Running     0
1h
polaris-consul-consul-server-0              1/1     Running     0
1h
polaris-consul-consul-server-1              1/1     Running     0
1h
polaris-consul-consul-server-2              1/1     Running     0
1h
polaris-keycloak-0                          1/1     Running     2 (1h
```

```
ago)      1h
polaris-keycloak-1                            1/1     Running     0
1h
polaris-keycloak-db-0                         1/1     Running     0
1h
polaris-keycloak-db-1                         1/1     Running     0
1h
polaris-keycloak-db-2                         1/1     Running     0
1h
polaris-mongodb-0                             1/1     Running     0
1h
polaris-mongodb-1                             1/1     Running     0
1h
polaris-mongodb-2                             1/1     Running     0
1h
polaris-ui-66476dcf87-f6s8j                   1/1     Running     0
1h
polaris-ui-66476dcf87-ztjk7                   1/1     Running     0
1h
polaris-vault-0                               1/1     Running     0
1h
polaris-vault-1                               1/1     Running     0
1h
polaris-vault-2                               1/1     Running     0
1h
public-metrics-bfc4fc964-x4m79                1/1     Running     0
1h
storage-backend-metrics-7dbb88d4bc-g78cj      1/1     Running     0
1h
storage-provider-5969b5df5-hjvcm              1/1     Running     0
1h
storage-provider-5969b5df5-r79ld              1/1     Running     0
1h
task-service-5fc9dc8d99-4q4f4                 1/1     Running     0
1h
task-service-5fc9dc8d99-8l5zl                 1/1     Running     0
1h
task-service-task-purge-28485735-fdzkd        1/1     Running     0
12m
telegraf-ds-2rgm4                             1/1     Running     0
1h
telegraf-ds-4qp6r                             1/1     Running     0
1h
telegraf-ds-77frs                             1/1     Running     0
1h
telegraf-ds-bc725                             1/1     Running     0
```

```
1h
telegraf-ds-cvmxf                              1/1     Running     0
1h
telegraf-ds-tqzgj                              1/1     Running     0
1h
telegraf-rs-5wtd8                              1/1     Running     0
1h
telemetry-service-6747866474-5djnc             1/1     Running     0
1h
telemetry-service-6747866474-thb7r             1/1     Running     1 (1h
ago)      1h
tenancy-5669854fb6-gzdzf                       1/1     Running     0
1h
tenancy-5669854fb6-xvsm2                       1/1     Running     0
1h
traefik-8f55f7d5d-4lgfw                        1/1     Running     0
1h
traefik-8f55f7d5d-j4wt6                        1/1     Running     0
1h
traefik-8f55f7d5d-p6gcq                        1/1     Running     0
1h
trident-svc-7cb5bb4685-54cnq                   1/1     Running     0
1h
trident-svc-7cb5bb4685-b28xh                   1/1     Running     0
1h
vault-controller-777b9bbf88-b5bqt              1/1     Running     0
1h
vault-controller-777b9bbf88-fdfd8              1/1     Running     0
1h
```

4. (선택 사항) 을(를) 확인합니다 `acc-operator` 진행 상황을 모니터링하기 위한 로그:

```
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 를 통해 등록을 다시 시도할 수 있습니다 "UI에서 클러스터 워크플로우를 추가합니다" API를 사용합니다.

5. 모든 Pod가 실행되면 설치가 성공적으로 완료되었는지 확인합니다 (`READY` 있습니다 `True`) 및 Astra Control Center에 로그인할 때 사용할 초기 설정 암호를 받습니다.

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

응답:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   24.02.0-69
10.111.111.111   True
```

UUID 값을 복사합니다. 암호는 ACC-, UUID 값(ACC-[UUID]), 이 예에서는 ACC-9aa5faaaaaaaud-4214-4cb7-9976-5d8b4c0ce27f)입니다.

부하 분산을 위한 수신 설정

서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 컨트롤러를 설정할 수 있습니다. 이 절차에서는 기본값을 사용한 경우 수신 컨트롤러에 대한 설정 예제를 제공합니다 `ingressType: "Generic"` Astra Control Center 사용자 지정 리소스 (`astra_control_center.yaml`)를 클릭합니다. 지정한 경우 이 절차를 사용할 필요가 없습니다 `ingressType: "AccTraefik"` Astra Control Center 사용자 지정 리소스 (`astra_control_center.yaml`)를 클릭합니다.

Astra Control Center가 구축된 후 Astra Control Center가 URL로 노출되도록 수신 컨트롤러를 구성해야 합니다.

설치 단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다. Astra Control Center는 다양한 수신 컨트롤러 유형을 지원합니다. 이러한 설정 절차는 일반적인 수신 컨트롤러 유형의 예를 제공합니다.

시작하기 전에

- 필수 요소입니다 "수신 컨트롤러" 이미 배포되어 있어야 합니다.
- 를 클릭합니다 "수신 클래스" 수신 컨트롤러에 해당하는 컨트롤러가 이미 생성되어야 합니다.

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 'istio-ingress.YAML' 파일에 제공된 'pec.tls.secretName'과 일치해야 합니다.

4. 에 수신 리소스를 배포합니다 `netapp-acc` (또는 사용자 지정 이름) 스키마에 대해 v1 리소스 형식을 사용하는 네임스페이스입니다 (`istio-Ingress.yaml` 이 예에서 사용됨):

```
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: [netapp-acc or custom namespace]
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. 변경 사항 적용:

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

6. 수신 상태를 점검하십시오.

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

응답:

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

7. Astra Control Center 설치를 완료합니다.

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) 스키마에 대해 v1 리소스 형식을 사용하는 네임스페이스입니다 (`nginx-Ingress.yaml` 이 예에서 사용됨):

```
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 address>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

3. 변경 사항 적용:

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

Nginx 컨트롤러를 이 아닌 배포로 설치하는 것이 좋습니다 `daemonSet`.

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 UI에 로그인합니다

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

단계

1. 브라우저에서 FQDN(을 포함)을 입력합니다 `https://` 접두사)를 입력합니다 `astraAddress` 에 있습니다 `astra_control_center.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.yaml` CR Astra Control Center를 설치했습니다를 누른 다음 초기 설치 암호를 입력합니다 (`ACC-[UUID]`)를 클릭합니다.

잘못된 암호를 세 번 입력하면 15분 동안 관리자 계정이 잠깁니다.

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

첫 번째 로그인인 경우 암호를 잊어버리고 다른 관리 사용자 계정이 아직 생성되지 않은 경우 에 문의하십시오 "NetApp 지원" 비밀번호 복구 지원을 위해.

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

설치 문제를 해결합니다

서비스 중 '오류' 상태인 서비스가 있으면 로그를 검사할 수 있습니다. 400 ~ 500 범위의 API 응답 코드를 찾습니다. 이는 고장이 발생한 장소를 나타냅니다.

옵션

- Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

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

- Astra Control Center CR의 출력을 확인하려면:

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

대체 설치 절차

- * Red Hat OpenShift OperatorHub를 사용하여 설치 *: 사용 "대체 절차" OperatorHub를 사용하여 OpenShift에 Astra Control Center를 설치하려면
- * Cloud Volumes ONTAP 백엔드를 사용하여 퍼블릭 클라우드에 설치 *: 사용 "수행할 수 있습니다" AWS(Amazon Web Services), GCP(Google Cloud Platform) 또는 Cloud Volumes ONTAP 스토리지 백엔드가 있는 Microsoft Azure에 Astra Control Center를 설치하려면 다음을 수행합니다.

다음 단계

- (선택 사항) 환경에 따라 사후 설치를 완료합니다 "구성 단계".
- "Astra Control Center를 설치하고 UI에 로그인하여 암호를 변경하면 라이센스를 설정하고, 클러스터를 추가하고, 인증을 활성화하고, 스토리지를 관리하고, 버킷을 추가할 수 있습니다".

외부 인증서 관리자를 구성합니다

Kubernetes 클러스터에 이미 인증 관리자가 있는 경우, Astra Control Center에서 자체 인증 관리자를 설치하지 않도록 몇 가지 필수 단계를 수행해야 합니다.

단계

1. 인증서 관리자가 설치되었는지 확인합니다.

```
kubectl get pods -A | grep 'cert-manager'
```

샘플 반응:

```
cert-manager   essential-cert-manager-84446f49d5-sf2zd              1/1
Running    0      6d5h
cert-manager   essential-cert-manager-cainjector-66dc99cc56-9ldmt   1/1
Running    0      6d5h
cert-manager   essential-cert-manager-webhook-56b76db9cc-fjqrq      1/1
Running    0      6d5h
```

2. 에 대한 인증서/키 쌍을 생성합니다 `astraAddress` FQDN:

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

샘플 반응:

```
Generating a 2048 bit RSA private key
..................+++
.......................................+++
writing new private key to 'tls.key'
```

3. 이전에 생성된 파일을 사용하여 암호 생성:

```
kubectl create secret tls selfsigned-tls --key tls.key --cert tls.crt -n
<cert-manager-namespace>
```

샘플 반응:

```
secret/selfsigned-tls created
```

4. 을 생성합니다 `ClusterIssuer` 정확히 * 인 파일 다음은 같지만 가 있는 네임스페이스 위치가 포함되어 있습니다 `cert-manager` Pod가 설치된 경우:

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: astra-ca-clusterissuer
  namespace: <cert-manager-namespace>
spec:
  ca:
    secretName: selfsigned-tls
```

```
kubectl apply -f ClusterIssuer.yaml
```

샘플 반응:

```
clusterissuer.cert-manager.io/astra-ca-clusterissuer created
```

5. 를 확인합니다 `ClusterIssuer` 이(가) 올바르게 나타납니다. `Ready` 은(는) 이어야 합니다 `True` 계속 진행하려면:

```
kubectl get ClusterIssuer
```

샘플 반응:

```
NAME                        READY   AGE
astra-ca-clusterissuer      True    9s
```

6. 를 완료합니다 "Astra Control Center 설치 프로세스". A가 있습니다 "Astra Control Center 클러스터 YAML에 필요한 구성 단계" 인증서 관리자가 외부에 설치되었음을 나타내기 위해 CRD 값을 변경합니다. Astra Control Center가 외부 인증서 관리자를 인식하도록 설치 중에 이 단계를 완료해야 합니다.

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.

시작하기 전에

- * 환경 필수 조건 충족 *: "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".

  Astra Control Center를 세 번째 고장 도메인 또는 보조 사이트에 배포합니다. 앱 복제 및 원활한 재해 복구에 권장됩니다.

- * 건강한 클러스터 운영자 및 API 서비스 보장 *:
  - OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다.

    ```
    oc get clusteroperators
    ```

  - OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다.

    ```
    oc get apiservices
    ```

- * 라우팅 가능한 FQDN *: 사용하려는 Astra FQDN을 클러스터로 라우팅할 수 있습니다. 즉, 내부 DNS 서버에 DNS 항목이 있거나 이미 등록된 코어 URL 경로를 사용하고 있는 것입니다.
- * OpenShift 권한 얻기 *: Red Hat OpenShift Container Platform에 대한 모든 필수 권한과 액세스 권한이 있어야 설명된 설치 단계를 수행할 수 있습니다.
- * 인증서 관리자 구성 *: 클러스터에 이미 인증서 관리자가 있는 경우 일부 작업을 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하지 않습니다. 기본적으로 Astra Control Center는 설치 중에 자체 인증서 관리자를 설치합니다.

- * Kubernetes Ingress 컨트롤러 설정 *: 클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes Ingress 컨트롤러가 있는 경우 Astra Control Center에서 사용하도록 설정해야 합니다.

  a. 연산자 네임스페이스 만들기:

  ```
  oc create namespace netapp-acc-operator
  ```

  b. "설정을 완료합니다" 수신 컨트롤러 유형에 적합합니다.

- * (ONTAP SAN 드라이버만 해당) 다중 경로 사용 *: ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

다음 사항도 고려해야 합니다.

- * NetApp Astra Control 이미지 레지스트리에 액세스 *:

  NetApp 이미지 레지스트리에서 Astra Control Provisioner와 같은 Astra Control의 설치 이미지 및 기능 개선 사항을 가져올 수 있습니다.

  a. 레지스트리에 로그인해야 하는 Astra Control 계정 ID를 기록합니다.

     계정 ID는 Astra Control Service 웹 UI에서 확인할 수 있습니다. 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API 액세스 * 를 선택한 후 계정 ID를 기록합니다.

  b. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.

  c. Astra Control 레지스트리에 로그인합니다.

  ```
  docker login cr.astra.netapp.io -u <account-id> -p <api-token>
  ```

- * 보안 통신을 위한 서비스 메시를 설치합니다 * : Astra Control 호스트 클러스터 통신 채널은 을 사용하여 보안을 유지하는 것이 좋습니다 "지원되는 서비스 메시입니다".

  > Astra Control Center를 서비스 메시와 통합하는 작업은 Astra Control Center 중에만 수행할 수 있습니다 "설치" 그리고 이 과정에 독립적이지 않습니다. 메시에서 메시되지 않은 환경으로 다시 변경하는 것은 지원되지 않습니다.

  Istio 서비스 메시를 사용하려면 다음을 수행해야 합니다.

  - 를 추가합니다 `istio-injection:enabled` Astra Control Center를 구축하기 전에 Astra 네임스페이스에 레이블을 지정합니다.
  - 를 사용합니다 `Generic` 수신 설정 에 대한 대체 침입을 제공합니다 "외부 부하 균형".
  - Red Hat OpenShift 클러스터의 경우 을 정의해야 합니다 `NetworkAttachmentDefinition` 연결된 모든 Astra Control Center 네임스페이스에서 (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` 응용 프로그램 클러스터 또는 대체된 사용자 지정 네임스페이스의 경우).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

단계

- Astra Control Center를 다운로드하고 압축을 풉니다
- 로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다
- 운영자 설치 페이지를 찾으십시오
- 운전자를 설치합니다
- Astra Control Center를 설치합니다

> Astra Control Center 운영자를 삭제하지 마십시오(예: `kubectl delete -f astra_control_center_operator_deploy.yaml`) 포드가 삭제되지 않도록 Astra Control Center 설치 또는 작동 중에 언제든지.

Astra Control Center를 다운로드하고 압축을 풉니다

다음 위치 중 하나에서 Astra Control Center 이미지를 다운로드하십시오.

- * Astra 컨트롤 서비스 이미지 레지스트리 *: Astra 컨트롤 센터 이미지에 로컬 레지스트리를 사용하지 않거나 NetApp Support 사이트에서 번들 다운로드보다 이 방법을 선호하는 경우 이 옵션을 사용합니다.
- * NetApp Support 사이트 *: Astra 컨트롤 센터 이미지와 함께 로컬 레지스트리를 사용하는 경우 이 옵션을 사용합니다.

Astra Control 이미지 레지스트리

1. Astra Control Service에 로그인합니다.
2. 대시보드에서 * Astra Control의 자가 관리형 인스턴스 배포 * 를 선택합니다.
3. 지침에 따라 Astra Control 이미지 레지스트리에 로그인하고 Astra Control Center 설치 이미지를 가져온 다음 이미지를 추출합니다.

NetApp Support 사이트

1. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장되지만 선택 사항) Astra Control Center용 인증서 및 서명 번들을 다운로드합니다 (`astra-control-center-certs-[version].tar.gz`)를 클릭하여 번들 서명을 확인합니다.

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

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

출력이 표시됩니다 `Verified OK` 확인 성공 후.

3. Astra Control Center 번들에서 이미지를 추출합니다.

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

로컬 레지스트리를 사용하는 경우 추가 단계를 완료합니다

Astra Control Center 번들을 로컬 레지스트리에 푸시하려는 경우 NetApp Astra kubectl 명령줄 플러그인을 사용해야 합니다.

NetApp Astra kubtl 플러그인을 설치합니다

최신 NetApp Astra kubectl 명령줄 플러그인을 설치하려면 다음 단계를 완료하십시오.

시작하기 전에

NetApp은 다양한 CPU 아키텍처 및 운영 체제에 대한 플러그인 바이너리를 제공합니다. 이 작업을 수행하기 전에 사용 중인 CPU 및 운영 체제를 알아야 합니다.

이전 설치에서 이미 플러그인을 설치한 경우 "최신 버전이 있는지 확인하십시오" 다음 단계를 수행하기 전에

단계

1. 사용 가능한 NetApp Astra kubectl 플러그인 바이너리를 나열하고 운영 체제 및 CPU 아키텍처에 필요한 파일 이름을 적어 주십시오.

kubbeck 플러그인 라이브러리는 tar 번들의 일부이며 폴더에 압축이 풀립니다 `kubectl-astra`.

```
ls kubectl-astra/
```

2. 올바른 바이너리를 현재 경로로 이동하고 이름을 로 변경합니다 `kubectl-astra`:

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

레지스트리에 이미지를 추가합니다

1. Astra Control Center 번들을 로컬 레지스트리로 푸시하려는 경우 컨테이너 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

a. 타볼의 루트 디렉토리로 변경합니다. 가 표시됩니다 `acc.manifest.bundle.yaml` 파일 및 다음 디렉토리:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. Astra Control Center 이미지 디렉토리의 패키지 이미지를 로컬 레지스트리에 밀어 넣습니다. 를 실행하기 전에 다음 대체 작업을 수행합니다 `push-images` 명령:

- <BUNDLE_FILE>를 Astra Control 번들 파일의 이름으로 바꿉니다 (`acc.manifest.bundle.yaml`)를 클릭합니다.
- <MY_FULL_REGISTRY_PATH>를 Docker 저장소의 URL로 바꿉니다. 예를 들어, "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>".
- <MY_REGISTRY_USER>를 사용자 이름으로 바꿉니다.
- <MY_REGISTRY_TOKEN>를 레지스트리에 대한 인증된 토큰으로 바꿉니다.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

팟맨

a. 타볼의 루트 디렉토리로 변경합니다. 이 파일과 디렉토리가 표시됩니다.

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 레지스트리에 로그인합니다.

```
podman login <YOUR_REGISTRY>
```

c. 사용하는 Podman 버전에 맞게 사용자 지정된 다음 스크립트 중 하나를 준비하고 실행합니다. <MY_FULL_REGISTRY_PATH>를 모든 하위 디렉토리가 포함된 리포지토리의 URL로 대체합니다.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

레지스트리 구성에 따라 스크립트가 만드는 이미지 경로는 다음과 같아야 합니다.

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 디렉토리를 변경합니다.

```
cd manifests
```

운영자 설치 페이지를 찾으십시오

1. 운영자 설치 페이지에 액세스하려면 다음 절차 중 하나를 완료하십시오.

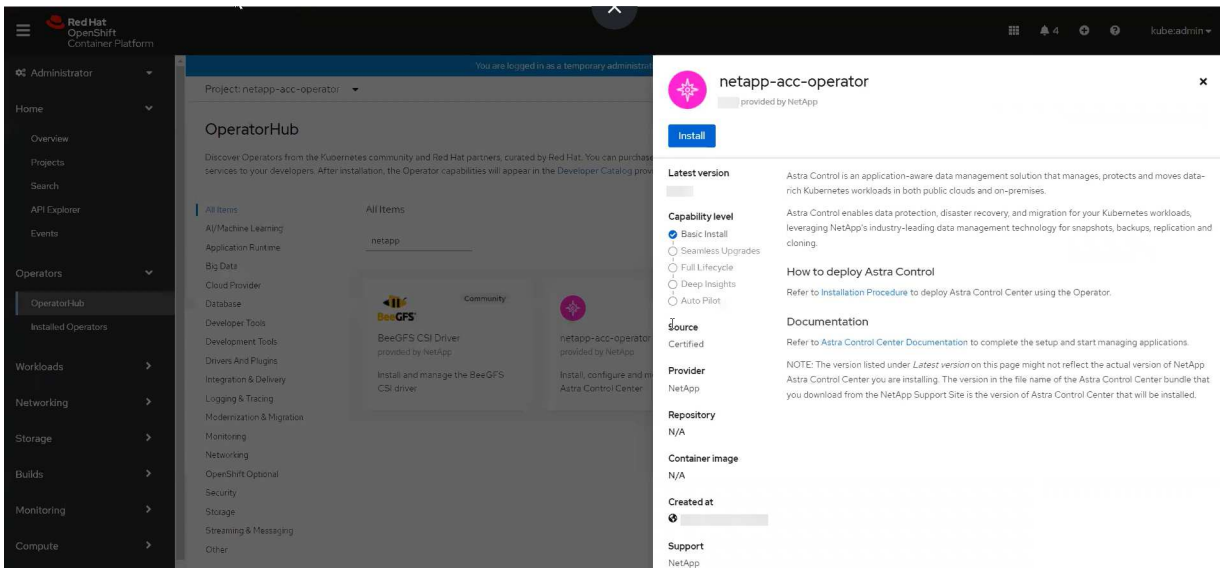
Red Hat OpenShift 웹 콘솔

a. OpenShift Container Platform UI에 로그인합니다.

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

> 이 연산자를 사용하여 현재 버전의 Astra Control Center에만 업그레이드할 수 있습니다.

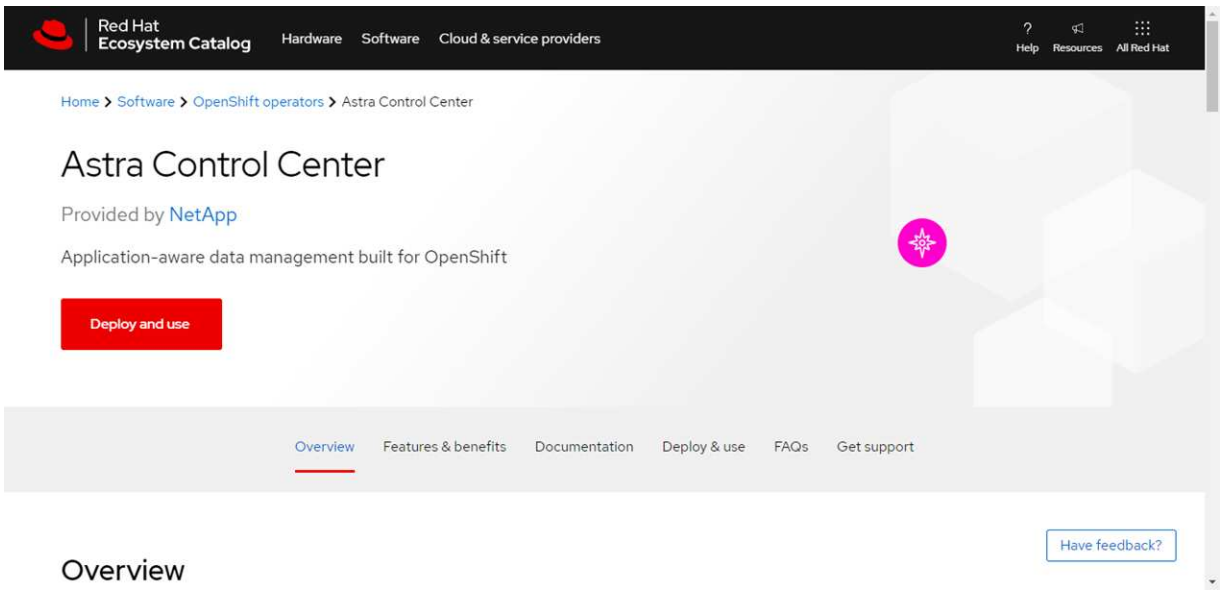
c. 을(를) 검색합니다 `netapp-acc` NetApp Astra Control Center 운영자를 선택합니다.



Red Hat 에코시스템 카탈로그

a. NetApp Astra Control Center를 선택합니다 "운영자".

b. 배포 및 사용 * 을 선택합니다.



운전자를 설치합니다

1. Install Operator * 페이지를 완료하고 운영자를 설치합니다.

   운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

   a. 운영자 설치의 일부로 운영자 네임스페이스 또는 'NetApp-acc-operator' 네임스페이스가 자동으로 생성됩니다.

   b. 수동 또는 자동 승인 전략을 선택합니다.

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

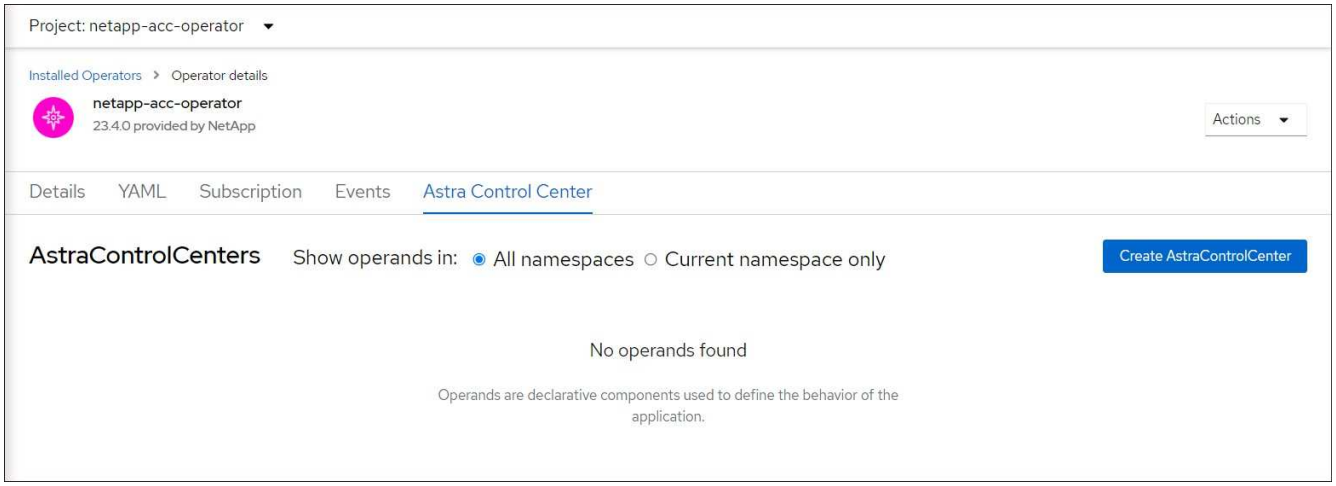
   c. 설치 * 를 선택합니다.

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

2. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

Astra Control Center를 설치합니다

1. Astra Control Center 운영자의 * Astra Control Center * 탭에 있는 콘솔에서 * Create AstraControlCenter * 를 선택합니다.



2. 'Create AstraControlCenter' 양식 필드를 작성합니다.

   a. Astra Control Center 이름을 유지하거나 조정합니다.

   b. Astra Control Center에 대한 레이블을 추가합니다.

   c. 자동 지원을 활성화 또는 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.

   d. Astra Control Center FQDN 또는 IP 주소를 입력합니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

   e. Astra Control Center 버전(예: 24.02.0-69)을 입력합니다.

   f. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.

   g. 의 볼륨 재확보 정책을 선택합니다 `Retain`, `Recycle`, 또는 `Delete`. 기본값은 입니다 `Retain`.

h. 설치의 배율 크기를 선택합니다.

> 기본적으로 Astra는 HA(High Availability)를 사용합니다. `scaleSize` 의 `Medium`즉, `HA`에서 대부분의 서비스를 구축하고 이중화를 위해 여러 복제본을 배포합니다. 와 함께 `scaleSize` 현재 `Small`Astra는 소비를 줄이기 위한 필수 서비스를 제외한 모든 서비스의 복제본 수를 줄일 것입니다.

i. 수신 유형을 선택합니다.

- * 일반 * (`ingressType: "Generic"`) (기본값)

  다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 구축한 후 를 구성해야 합니다 "수신 컨트롤러" URL을 사용하여 Astra Control Center를 표시합니다.

- * AccTraefik * (`ingressType: "AccTraefik"`)

  수신 컨트롤러를 구성하지 않으려는 경우 이 옵션을 사용하십시오. 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".

a. Image Registry*에서 로컬 레지스트리를 구성하지 않은 경우 기본값을 사용합니다. 로컬 레지스트리의 경우 이 값을 이전 단계에서 이미지를 푸시한 로컬 이미지 레지스트리 경로로 바꿉니다. 들어가지마 `http://` 또는 `https://` 를 입력합니다.

b. 인증이 필요한 이미지 레지스트리를 사용하는 경우 이미지 암호를 입력합니다.

> 인증이 필요한 레지스트리를 사용하는 경우 클러스터에 암호를 생성합니다.

c. 관리자의 이름을 입력합니다.

d. 리소스 확장을 구성합니다.

e. 기본 스토리지 클래스를 제공합니다.

> 기본 스토리지 클래스가 구성된 경우 기본 주석이 있는 유일한 스토리지 클래스인지 확인합니다.

f. CRD 처리 기본 설정을 정의합니다.

3. YAML 보기를 선택하여 선택한 설정을 검토합니다.
4. Create를 선택합니다.

레지스트리 암호를 만듭니다

인증이 필요한 레지스트리를 사용하는 경우 OpenShift 클러스터에 암호를 생성하고 에 암호 이름을 입력합니다 `Create AstraControlCenter` 양식 필드.

1. Astra Control Center 운영자용 네임스페이스를 생성합니다.

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. 이 네임스페이스에 암호 만들기:

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

Astra Control은 Docker 레지스트리 비밀만 지원합니다.

3. 의 나머지 필드를 작성합니다 Create AstraControlCenter 양식 필드.

다음 단계

를 완료합니다 "나머지 단계" Astra Control Center가 성공적으로 설치되었는지 확인하려면 수신 컨트롤러(옵션)를 설정하고 UI에 로그인합니다. 또한 를 수행해야 합니다 "설정 작업" 설치 완료 후.

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 관리형 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 Amazon Route 53 항목이 필요합니다

AWS의 운영 환경 요구사항

Astra Control Center에는 AWS를 위한 다음과 같은 운영 환경이 필요합니다.

- Red Hat OpenShift Container Platform 4.11 ~ 4.13

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

Astra Control Center에는 환경의 리소스 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

> AWS 레지스트리 토큰이 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. AWS용 NetApp BlueXP를 구성합니다.
5. AWS용 Astra Control Center를 설치합니다.

IAM 권한이 충분한지 확인하십시오

RedHat OpenShift 클러스터와 NetApp BlueXP(이전의 Cloud Manager) 커넥터를 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 생성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스팅하기 위한 ECR(Elastic Container Registry)도 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

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

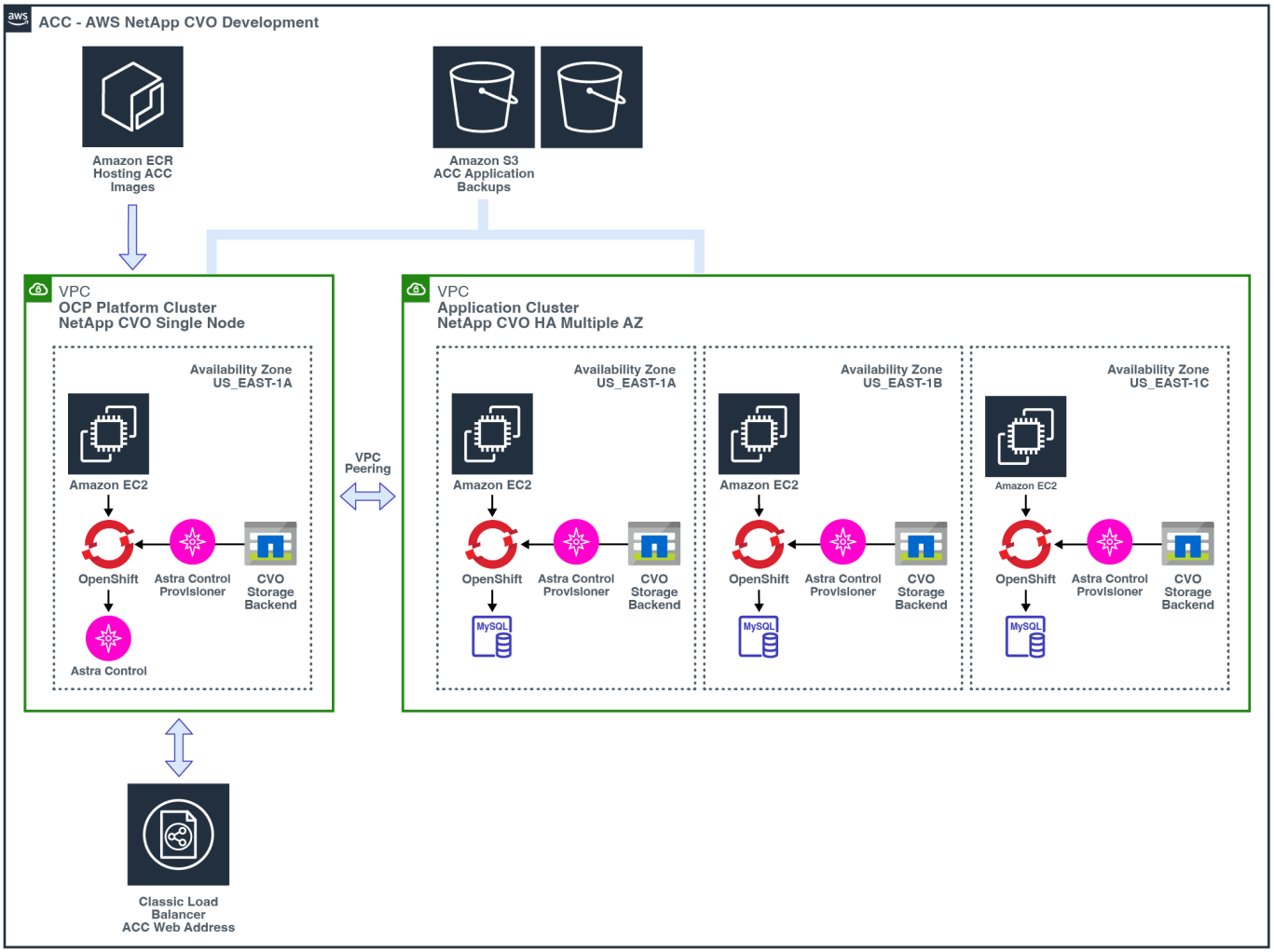
1. AWS 가상 네트워크를 생성합니다.
2. EC2 컴퓨팅 인스턴스를 검토합니다. 이는 AWS의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 AWS의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 AWS S3 버킷을 하나 이상 생성합니다.
5. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.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

      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.

   b. 정의된 레지스트리에 Astra Control Center 이미지를 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.



AWS용 **NetApp BlueXP**를 구성합니다

NetApp BlueXP(이전의 Cloud Manager)를 사용하여 작업 공간을 생성하고, AWS에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 다음을 참조하십시오.

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

단계

1. BlueXP에 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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.

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.
Astra Control Provisioner는 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.

AWS용 Astra Control Center를 설치합니다

표준을 따릅니다 "Astra Control Center 설치 지침".

AWS는 일반 S3 버킷 유형을 사용합니다.

Google Cloud Platform에 Astra Control Center를 구축합니다

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

GCP에 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) 4.11 ~ 4.13
- 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

GCP의 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 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

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 BlueXP for GCP를 구성합니다.
6. Astra Control Center for GCP를 설치합니다.

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 BlueXP(이전의 Cloud Manager) 커넥터를 설치할 수 있도록 충분한 IAM 역할 및 권한이 있는지 확인합니다.

을 참조하십시오 "초기 GCP 자격 증명 및 권한".

GCP를 구성합니다

다음으로, GCP를 구성하여 VPC를 생성하고, 컴퓨팅 인스턴스를 설정하고, Google Cloud Object Storage를 생성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스트하기 위한 Google 컨테이너 레지스트리를 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

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. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.
   a. Google Container Registry를 생성하여 Astra Control Center 이미지를 호스트합니다.
   b. 모든 Astra Control Center 이미지에 대해 Docker 푸시/풀용 Google Container Registry 액세스를 설정합니다.

      예: Astra Control Center 이미지는 다음 스크립트를 입력하여 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=acc.manifest.bundle.yaml
GCP_CR_REGISTRY=<target GCP image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 < acc.manifest.bundle.yaml
```

7. DNS 존 설정

NetApp BlueXP for GCP를 구성합니다

NetApp BlueXP(이전의 Cloud Manager)를 사용하여 작업 공간을 만들고, GCP에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 을 참조하십시오 "GCP에서 Cloud Volumes ONTAP 시작하기".

시작하기 전에

- 필요한 IAM 권한 및 역할을 사용하여 GCP 서비스 계정에 액세스합니다

단계

1. BlueXP에 자격 증명을 추가합니다. 을 참조하십시오 "GCP 계정 추가".
2. GCP용 커넥터를 추가합니다.
   a. 공급자로 "GCP"를 선택합니다.
   b. GCP 자격 증명을 입력합니다. 을 참조하십시오 "BlueXP에서 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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.
   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.
      Astra Control Provisioner는 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 for GCP를 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 Azure에 Astra Control Center를 구축합니다

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

Azure에 필요한 기능

Azure에 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) 4.11 ~ 4.13

- 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

Azure의 운영 환경 요구사항

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

Astra Control Center에는 환경의 리소스 요구사항 외에 특정 리소스가 필요하다. 을 참조하십시오 "Astra Control Center 운영 환경 요구 사항".

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. Azure용 NetApp BlueXP(이전의 Cloud Manager)를 구성합니다.
6. Azure용 Astra Control Center를 설치 및 구성합니다.

Azure에 RedHat OpenShift 클러스터를 설치합니다

첫 번째 단계는 Azure에 RedHat OpenShift 클러스터를 설치하는 것입니다.

설치 지침은 다음을 참조하십시오.

- "Azure에 OpenShift 클러스터 설치".
- "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 BlueXP 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 만들도록 Azure를 구성합니다. NetApp Astra 컨트롤 센터 이미지 레지스트리에 액세스할 수 없는 경우 Astra 컨트롤 센터 이미지를 호스팅하기 위한 Azure 컨테이너 레지스트리(ACR)도 생성하고 이미지를 이 레지스트리에 푸시해야 합니다.

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 스토리지 계정이 필요합니다.
6. 버킷 액세스에 필요한 암호를 생성합니다.
7. (선택 사항) NetApp 이미지 레지스트리에 액세스할 수 없는 경우 다음을 수행합니다.
   a. Azure 컨테이너 레지스트리(ACR)를 생성하여 Astra Control Center 이미지를 호스팅합니다.
   b. 모든 Astra Control Center 이미지에 대해 Docker 푸시/풀용 ACR 액세스를 설정합니다.
   c. 다음 스크립트를 사용하여 Astra Control Center 이미지를 이 레지스트리에 푸시합니다.

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

      - 예 *:

```
manifestfile=acc.manifest.bundle.yaml
AZ_ACR_REGISTRY=<target Azure ACR image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. DNS 존 설정

Azure용 **NetApp BlueXP(**이전의 **Cloud Manager)**를 구성합니다

BlueXP(이전의 Cloud Manager)를 사용하여 작업 영역을 만들고, Azure에 커넥터를 추가하고, 작업 환경을 생성하고, 클러스터를 가져옵니다.

BlueXP 설명서를 참조하여 다음 단계를 완료합니다. 을 참조하십시오 "Azure에서 BlueXP를 시작합니다".

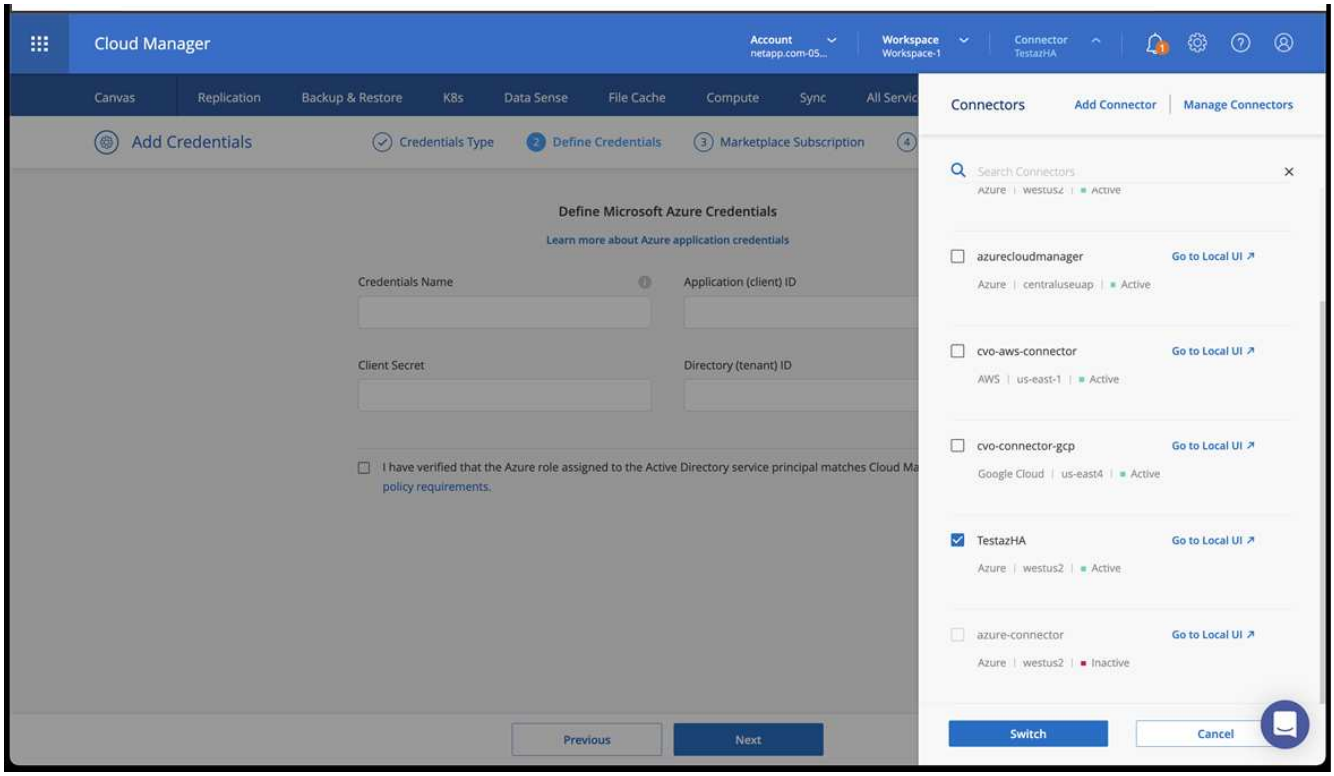
시작하기 전에

필요한 IAM 권한 및 역할을 사용하여 Azure 계정에 액세스합니다

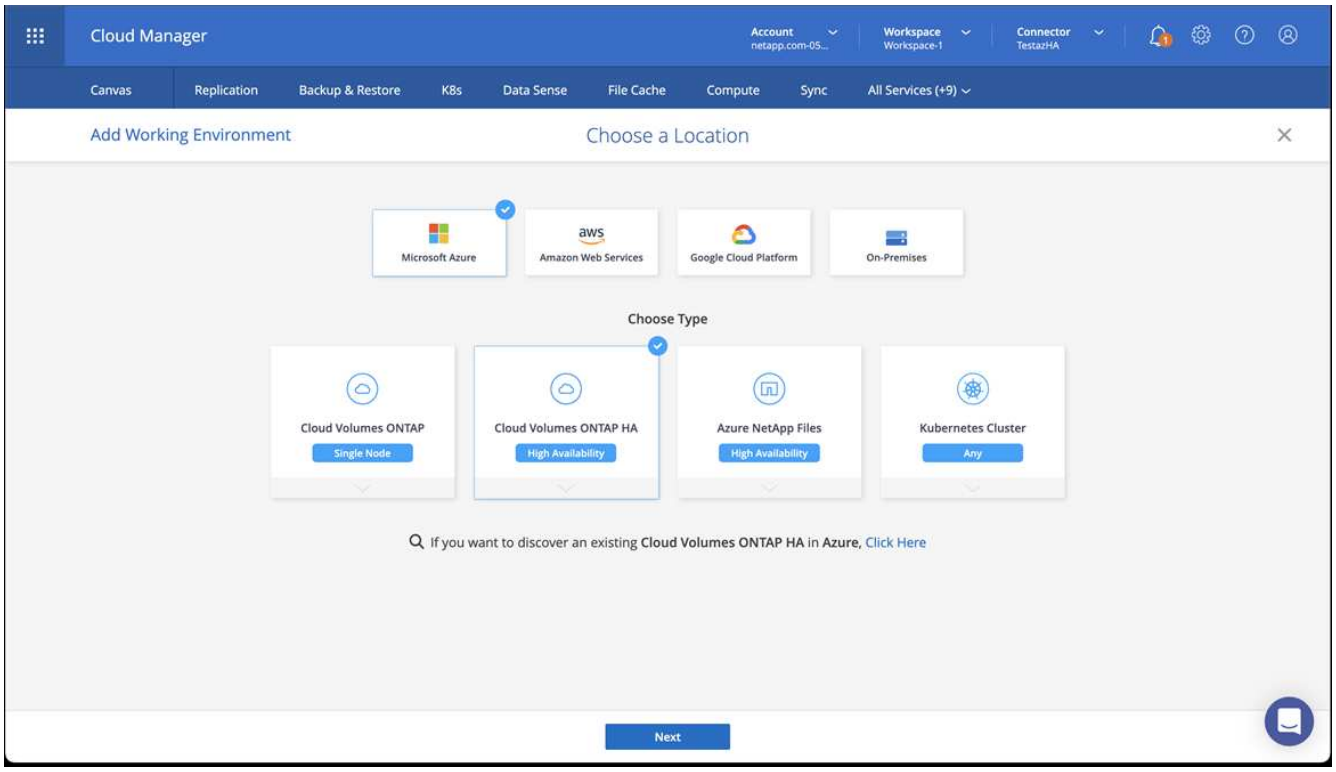
단계

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

      을 참조하십시오 "BlueXPr에서 커넥터 만들기".
3. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.

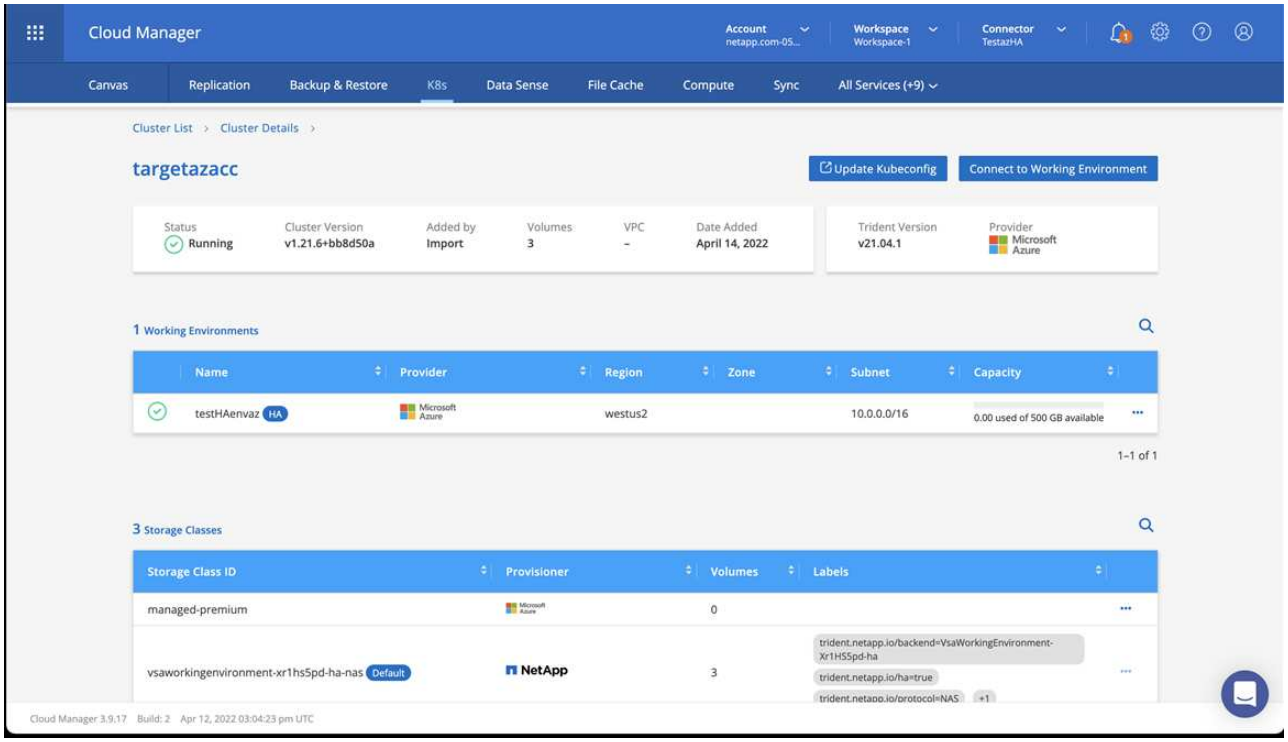


4. 클라우드 환경을 위한 작업 환경을 구축합니다.
   a. 위치: "Microsoft Azure".
   b. "Cloud Volumes ONTAP HA"를 입력합니다.



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 모서리에 Astra Control Provisioner 버전이 나와 있습니다.

   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.
Astra Control Provisioner는 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.

Azure용 Astra Control Center를 설치 및 구성합니다

Astra Control Center를 표준으로 설치합니다 "설치 지침".

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

설치 후 Astra Control Center를 구성합니다

환경에 따라 Astra Control Center를 설치한 후 추가 구성이 필요할 수 있습니다.

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. 이러한 방식으로 환경을 구성한 경우 Astra Control Center를 설치할 네임스페이스에서 해당 리소스를 제거해야 합니다.

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

단계

1. 에서 리소스 할당량을 가져옵니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스:

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

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 or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. 에서 제한 범위를 가져옵니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스:

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

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 or custom
namespace]
```

사용자 지정 **TLS** 인증서를 추가합니다

Astra Control Center는 자체 서명된 TLS 인증서를 수신 컨트롤러 트래픽(특정 구성에만 해당)과 웹 브라우저를 통한 웹 UI 인증에 사용합니다. 프로덕션 사용을 위해 기존의 자체 서명된 TLS 인증서를 제거하고 인증 기관(CA)에서 서명한 TLS 인증서로 교체해야 합니다.

> 자체 서명된 기본 인증서는 다음 두 가지 연결 유형에 사용됩니다.
>
> - Astra Control Center 웹 UI에 대한 HTTPS 연결
> - 수신 컨트롤러 트래픽(에만 해당 `ingressType: "AccTraefik"` 속성이 에 설정되었습니다 `astra_control_center.yaml` Astra Control Center 설치 중 파일)
>
> 기본 TLS 인증서를 교체하면 이러한 연결에 인증에 사용되는 인증서가 대체됩니다.

시작하기 전에

- Astra Control Center가 설치된 Kubernetes 클러스터

- 클러스터의 명령 셸에 대한 관리 액세스로 "kubbtl" 명령을 실행합니다
- 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. '<ACC-deployment-namespace>'를 Astra Control Center 배포 네임스페이스로 대체하여 현재 인증서와 연결된 TLS 암호를 찾습니다.

```
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>
```

명령줄을 사용하여 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 'pec.selfSigned` 값을 'pec.ca.secretName` 으로 변경하여 앞에서 생성한 TLS 암호를 참조합니다.

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

CRD:

```
#spec:
#  selfSigned: {}

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

3. 다음 명령 및 예제 출력을 사용하여 변경 사항이 올바르고 클러스터가 인증서를 검증할 준비가 되었는지 확인하고 "<ACC-deployment-namespace>"를 Astra Control Center 배포 네임스페이스로 대체합니다.

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

응답:

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

4. 다음 예를 사용하여 대괄호 <>의 자리 표시자 값을 적절한 정보로 대체하여 "certificate.yAML" 파일을 작성합니다.

> 이 예에서는 를 사용합니다 `dnsNames` Astra Control Center DNS 주소를 지정하는 속성입니다. Astra Control Center는 DNS 주소를 지정하는 CN(Common Name) 속성을 사용할 수 없습니다.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  <strong>name: <certificate-name></strong>
  namespace: <ACC-deployment-namespace>
spec:
  <strong>secretName: <certificate-secret-name></strong>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  <strong>- <astra.dnsname.example.com></strong> #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>
```

응답:

```
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. TLS 편집 다음 명령 및 예제를 사용하여 새 인증서 암호 이름을 가리키도록 CRD를 저장합니다. 대괄호 <>의 개체 틀 값을 적절한 정보로 바꿉니다

```
kubectl edit tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD:

```
...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>
```

8. 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.io -n <ACC-deployment-namespace>
```

CRD:

```
...
 tls:
    secretName: <certificate-secret-name>
```

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

Astra Control Center를 설정합니다

Astra Control Center에 대한 라이센스를 추가합니다

Astra Control Center를 설치할 때 포함된 평가판 라이센스가 이미 설치되어 있습니다. Astra Control Center를 평가하는 경우 이 단계를 건너뛸 수 있습니다.

Astra Control UI 또는 를 사용하여 새 라이센스를 추가할 수 있습니다 "Astra Control API를 참조하십시오".

Astra Control Center 라이센스는 Kubernetes CPU 유닛을 사용하여 CPU 리소스를 측정하고, 모든 관리되는 Kubernetes 클러스터의 작업자 노드에 할당된 CPU 리소스를 고려합니다. 라이센스는 vCPU 사용량을 기준으로 합니다. 라이선스 계산 방법에 대한 자세한 내용은 을 참조하십시오 "라이센싱".

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

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

시작하기 전에

- 새로 설치된 Astra Control Center 인스턴스에 액세스합니다.
- 관리자 역할 권한.
- A "NetApp 라이센스 파일" (NLF)

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.

평가판 라이센스가 있고 데이터를 AutoSupport로 전송하지 않는 경우, Astra Control Center에 장애가 발생할 경우 데이터 손실을 방지하기 위해 계정 ID를 저장해야 합니다.

Astra Control Provisioner를 활성화합니다

Astra Trident 버전 23.10 이상에는 라이센스를 보유한 Astra Control 사용자가 고급 스토리지 프로비저닝 기능에 액세스할 수 있도록 Astra Control Provisioner를 사용하는 옵션이 포함되어 있습니다. Astra Control Provisioner는 표준 Astra Trident CSI 기반 기능과 더불어 이 확장 기능을 제공합니다.

향후 Astra Control 업데이트에서 Astra Control Provisioner는 Astra Trident를 스토리지 프로비저닝 및 오케스트레이터로 대체하며 Astra Control을 사용하려면 필수입니다. 따라서 Astra Control 사용자가 Astra Control Provisioner를 활성화하는 것이 좋습니다. Astra Trident는 오픈 소스를 계속 유지하며, NetApp의 새로운 CSI 및 기타 기능으로 릴리즈, 유지, 지원 및 업데이트될 것입니다.

이 작업에 대해

Astra Control Center 사용이 허가된 사용자이고 Astra Control Provisioner 기능을 사용하려는 경우에는 이 절차를 따라야 합니다. Astra Trident 사용자가 Astra Control을 사용하지 않고 Astra Control Provisioner가 제공하는 추가 기능을 사용하려면 이 절차를 따라야 합니다.

각 사례에 대해 Astra Trident 24.02에서 Provisioner 기능이 기본적으로 활성화되어 있지 않으며 활성화되어야 합니다.

시작하기 전에

Astra Control Provisioner를 사용하도록 설정하려면 먼저 다음을 수행합니다.

Astra Control Center를 통해 사용자에게 **Astra Control Provisioner** 제공

- * Astra Control Center 라이센스 받기 *: 가 필요합니다 "Astra Control Center 라이센스" Astra Control Provisioner를 활성화하고 이 기능이 제공하는 기능에 액세스합니다.
- * Astra Control Center 23.10 이상 설치 또는 업그레이드 *: Astra Control과 함께 최신 Astra Control Provisioner 기능(24.02)을 사용하려면 최신 Astra Control Center 버전(24.02)이 필요합니다.
- * 클러스터에 AMD64 시스템 아키텍처가 있는지 확인 *: Astra Control Provisioner 이미지는 AMD64 및 ARM64 CPU 아키텍처 모두에서 제공되지만 Astra Control Center에서는 AMD64만 지원됩니다.
- * 레지스트리 액세스를 위한 Astra Control Service 계정 얻기 * : NetApp Support 사이트 대신 Astra Control 레지스트리를 사용하여 Astra Control Provisioner 이미지를 다운로드하려면 에 대한 등록을 완료하십시오 "Astra Control Service 계정". 양식을 작성하여 제출하고 BlueXP 계정을 생성하면 Astra Control Service 환영 이메일이 전송됩니다.
- * Astra Trident를 설치한 경우 해당 버전이 4개의 릴리즈 창 내에 있는지 확인 *: Astra Trident가 버전 24.02의 4개의 릴리즈 창 내에 있는 경우 Astra Control Provisioner를 사용하여 Astra Trident 24.02로 직접 업그레이드할 수 있습니다. 예를 들어, Astra Trident 23.04에서 24.02로 직접 업그레이드할 수 있습니다.

Astra Control Provisioner 전용 사용자

- * Astra Control Center 라이센스 받기 *: 가 필요합니다 "Astra Control Center 라이센스" Astra Control Provisioner를 활성화하고 이 기능이 제공하는 기능에 액세스합니다.
- * Astra Trident를 설치한 경우 해당 버전이 4개의 릴리즈 창 내에 있는지 확인 *: Astra Trident가 버전 24.02의 4개의 릴리즈 창 내에 있는 경우 Astra Control Provisioner를 사용하여 Astra Trident 24.02로 직접 업그레이드할 수 있습니다. 예를 들어, Astra Trident 23.04에서 24.02로 직접 업그레이드할 수 있습니다.
- * 레지스트리 액세스를 위한 Astra Control Service 계정 얻기 * : Astra Control Provisioner 이미지를 다운로드하려면 레지스트리에 액세스해야 합니다. 시작하려면 에 대한 등록을 완료하십시오 "Astra Control Service 계정". 양식을 작성하여 제출하고 BlueXP 계정을 생성하면 Astra Control Service 환영 이메일이 전송됩니다.

(1단계) **Astra Control Provisioner** 이미지를 가져옵니다

Astra Control Center 사용자는 Astra Control 레지스트리 또는 NetApp Support 사이트 방법을 사용하여 Astra Control Provisioner 이미지를 가져올 수 있습니다. Astra Control 없이 Astra Control Provisioner를 사용하려는 Astra Trident 사용자는 레지스트리 방법을 사용해야 합니다.

Astra Control 이미지 레지스트리

이 절차의 명령에 Docker 대신 Podman을 사용할 수 있습니다. Windows 환경을 사용하는 경우 PowerShell을 사용하는 것이 좋습니다.

1. NetApp Astra Control 이미지 레지스트리에 액세스:
   a. Astra Control Service 웹 UI에 로그온하여 페이지 오른쪽 상단의 그림 아이콘을 선택합니다.
   b. API 액세스 * 를 선택합니다.
   c. 계정 ID를 기록합니다.
   d. 같은 페이지에서 * API 토큰 생성 * 을 선택하고 API 토큰 문자열을 클립보드에 복사하여 편집기에 저장합니다.
   e. 원하는 방법을 사용하여 Astra Control 레지스트리에 로그인합니다.

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (사용자 지정 레지스트리에만 해당) 이미지를 사용자 지정 레지스트리로 이동하려면 다음 단계를 수행하십시오. 레지스트리를 사용하지 않는 경우 의 Trident 운영자 단계를 따르십시오 "다음 섹션을 참조하십시오".
   a. 레지스트리에서 Astra Control Provisioner 이미지를 가져옵니다.

가져온 이미지는 여러 플랫폼을 지원하지 않으며 Linux AMD64와 같이 이미지를 가져온 호스트와 동일한 플랫폼만 지원합니다.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform <cluster platform>
```

예:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform linux/amd64
```

   a. 이미지에 태그 지정:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0
```

b. 이미지를 사용자 지정 레지스트리에 푸시합니다.

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

> 다음 Docker 명령을 실행하는 대신 크레인 복사본을 사용할 수 있습니다.
> `crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0`

NetApp Support 사이트

1. Astra Control Provisioner 번들을 다운로드합니다 (`trident-acp-[version].tar`)를 선택합니다 "Astra Control Center 다운로드 페이지".
2. (권장 사항이지만 선택 사항) Astra Control Center(Astra-control-center-certs-[version].tar.gz)용 인증서 및 서명 번들을 다운로드하여 trident-acp-[version] tar 번들의 서명을 확인하십시오.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenterDockerImages-public.pub -signature certs/trident-acp-[version].tar.sig trident-acp-[version].tar
```

3. Astra Control Provisioner 이미지 로드:

```
docker load < trident-acp-24.02.0.tar
```

응답:

```
Loaded image: trident-acp:24.02.0-linux-amd64
```

4. 이미지에 태그 지정:

```
docker tag trident-acp:24.02.0-linux-amd64 <my_custom_registry>/trident-acp:24.02.0
```

5. 이미지를 사용자 지정 레지스트리에 푸시합니다.

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

(2단계) Astra Trident에서 Astra Control Provisioner를 사용하도록 설정합니다

원래 설치 방법으로 를 사용했는지 확인합니다 "연산자(수동 또는 Helm 사용) 또는 tridentctl" 그리고 원래 방법에 따라 적절한 단계를 완료합니다.

Astra Trident 운영자

1. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다".
2. Astra Trident를 아직 설치하지 않았거나 원본 Astra Trident 구축에서 연산자를 제거한 경우 다음 단계를 완료하십시오.
   a. CRD 생성:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.y
aml
```

   b. 트라이덴트 네임스페이스를 만듭니다 (`kubectl create namespace trident`) 또는 트리덴트 네임스페이스가 여전히 존재하는지 확인합니다 (`kubectl get all -n trident`)를 클릭합니다. 네임스페이스가 제거된 경우 다시 만듭니다.
3. Astra Trident를 24.02.0으로 업데이트:

> Kubernetes 1.24 이하 버전을 실행하는 클러스터의 경우, 를 사용합니다 `bundle_pre_1_25.yaml`. Kubernetes 1.25 이상을 실행하는 클러스터의 경우 를 사용합니다 `bundle_post_1_25.yaml`.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

4. Astra Trident가 실행 중인지 확인합니다.

```
kubectl get torc -n trident
```

   응답:

```
NAME      AGE
trident   21m
```

5.] 비밀을 사용하는 레지스트리가 있는 경우 Astra Control Provisioner 이미지를 가져오는 데 사용할 비밀을 만듭니다.

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

6. TridentOrchestrator CR을 편집하고 다음과 같이 편집합니다.

```
kubectl edit torc trident -n trident
```

a. Astra Trident 이미지에 대한 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리에서 가져옵니다 (`tridentImage: <my_custom_registry>/trident:24.02.0` 또는 `tridentImage: netapp/trident:24.02.0`)를 클릭합니다.

b. Astra Control Provisioner를 활성화합니다 (`enableACP: true`)를 클릭합니다.

c. Astra Control Provisioner 이미지의 사용자 지정 레지스트리 위치를 설정하거나 Astra Control 레지스트리에서 가져옵니다 (`acpImage: <my_custom_registry>/trident-acp:24.02.0` 또는 `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`)를 클릭합니다.

d. 를 설정했는지 확인합니다 이미지 풀 암호 이 절차의 앞부분에서 여기에서 설정할 수 있습니다 (`imagePullSecrets: - <secret_name>`)를 클릭합니다. 이전 단계에서 설정한 것과 동일한 이름 암호 이름을 사용합니다.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

7. 파일을 저장하고 종료합니다. 배포 프로세스가 자동으로 시작됩니다.
8. 운영자, 배포 및 복제 세트가 생성되었는지 확인합니다.

```
kubectl get all -n trident
```

Kubernetes 클러스터에는 운영자의 인스턴스 * 하나가 있어야 합니다. Astra Trident 연산자를 여러 번 구축해서는 안 됩니다.

9. 를 확인합니다 `trident-acp` 컨테이너가 실행 중이며 `acpVersion` 있습니다 `24.02.0` 의 상태입니다 `Installed`:

```
kubectl get torc -o yaml
```

응답:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl 을 선택합니다

1. "Astra Trident 설치 프로그램을 다운로드하여 압축을 풉니다".
2. "기존 Astra Trident가 있는 경우 이를 호스팅하는 클러스터에서 제거합니다".
3. Astra Control Provisioner를 사용하도록 설정된 Astra Trident를 설치합니다 (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

4. Astra Control Provisioner가 활성화되었는지 확인합니다.

```
./tridentctl -n trident version
```

응답:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+---------------
+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. | +----------------
+----------------+-------------+
```

헬름

1. Astra Trident 23.07.1 이하를 설치한 경우 "설치 제거" 작업자 및 기타 구성품
2. Kubernetes 클러스터에서 1.24 이전 버전을 실행 중인 경우 psp:

```
kubectl delete psp tridentoperatorpod
```

3. Astra Trident Helm 리포지토리를 추가합니다.

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

4. 제어 차트 업데이트:

```
helm repo update netapp-trident
```

응답:

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

5. 영상을 나열합니다.

```
./tridentctl images -n trident
```

응답:

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-driver-
registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

6. 트라이덴트 - 운전자 24.02.0을 사용할 수 있는지 확인합니다.

```
helm search repo netapp-trident/trident-operator --versions
```

응답:

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

7. 사용 `helm install` 을 클릭하고 다음 설정을 포함하는 옵션 중 하나를 실행합니다.
   - 배포 위치의 이름입니다
   - Astra Trident 버전
   - Astra Control Provisioner 이미지의 이름
   - Provisioner를 활성화하는 플래그입니다
   - (선택 사항) 로컬 레지스트리 경로입니다. 로컬 레지스트리를 사용하는 경우, 을(를) 참조하십시오 "Trident 이미지" 하나의 레지스트리 또는 다른 레지스트리에 있을 수 있지만 모든 CSI 이미지는 동일한 레지스트리에 있어야 합니다.
   - Trident 네임스페이스

옵션

- 레지스트리가 없는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-acp:24.02.0
--set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- 하나 이상의 레지스트리에 있는 이미지

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

을 사용할 수 있습니다 `helm list` 이름, 네임스페이스, 차트, 상태, 앱 버전과 같은 설치 세부 정보를 검토하려면 수정본 번호.

Helm을 사용하여 Trident를 구축하는 데 문제가 있는 경우 다음 명령을 실행하여 Astra Trident를 완전히 제거합니다.

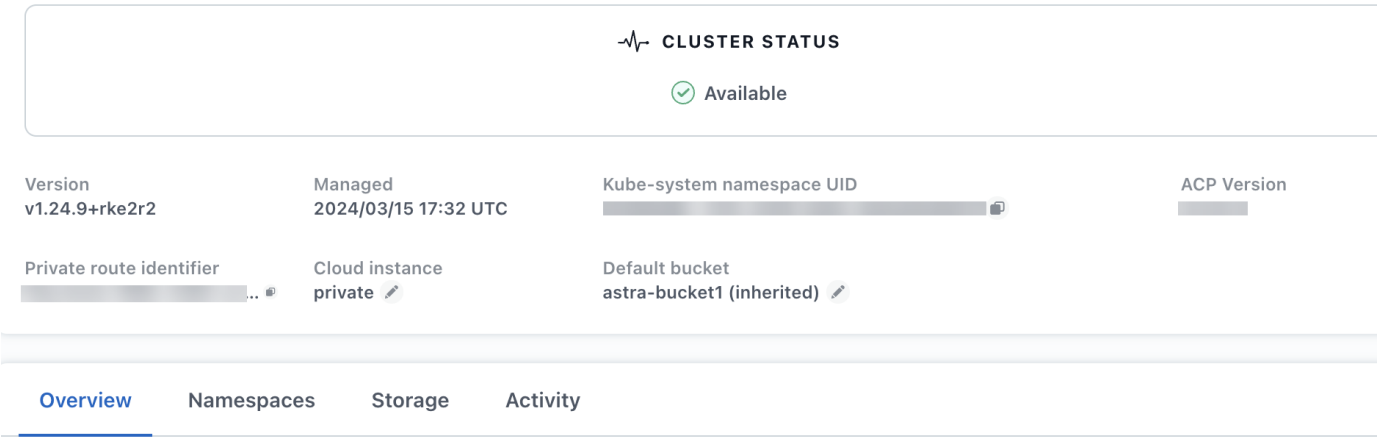
```
./tridentctl uninstall -n trident
```

- 하지 마십시오 * "Astra Trident CRD를 완전히 제거합니다" 설치 제거의 일부로 Astra Control Provisioner를 다시 활성화하려고 합니다.

결과

Astra Control Provisioner 기능이 활성화되어 있으며 실행 중인 버전에 제공되는 모든 기능을 사용할 수 있습니다.

(Astra Control Center 사용자만 해당) Astra Control Provisioner를 설치하면 Astra Control Center UI에서 Provisioner를 호스팅하는 클러스터에 가 표시됩니다 `ACP version` 을 사용하지 마십시오 `Trident version` 필드 및 현재 설치된 버전 번호



를 참조하십시오

- "Astra Trident 업그레이드 설명서"

Astra Control을 사용하여 클러스터 관리를 위한 환경을 준비합니다

클러스터를 추가하기 전에 다음 전제 조건이 충족되어야 합니다. 또한 자격 검사를 실행하여 클러스터를 Astra Control Center에 추가할 준비가 되었는지 확인하고 필요에 따라 kubeconfig 클러스터 역할을 생성해야 합니다.

Astra Control을 사용하면 환경 및 선호도에 따라 CR(Custom Resource) 또는 kubeconfig로 관리되는 클러스터를 추가할 수 있습니다.

시작하기 전에

- * 환경 필수 조건 충족 *: 사용자 환경이 충족됩니다 "구현할 수 있습니다" Astra Control Center의 경우
- * 작업자 노드 구성 *: 확인하십시오 "작업자 노드를 구성합니다" 클러스터에서 Pod가 백엔드 스토리지와 상호 작용할 수 있도록 적절한 스토리지 드라이버가 있어야 합니다.
- * PSA 제한 활성화 *: 클러스터에 Kubernetes 1.25 이상 클러스터의 표준인 Pod 보안 허용 적용이 활성화되어 있으면 이 네임스페이스에 PSA 제한을 활성화해야 합니다.
  - `netapp-acc-operator` 네임스페이스:

```
kubectl label --overwrite ns netapp-acc-operator pod-
security.kubernetes.io/enforce=privileged
```

- `netapp monitoring` 네임스페이스:

```
kubectl label --overwrite ns netapp-monitoring pod-
security.kubernetes.io/enforce=privileged
```

- * ONTAP credentials *: Astra Control Center를 사용하여 앱을 백업 및 복원하려면 ONTAP 시스템에 ONTAP 자격 증명과 고급 사용자 및 사용자 ID가 설정되어 있어야 합니다.

  ONTAP 명령줄에서 다음 명령을 실행합니다.

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

- * kubeconfig-managed cluster requirements *: 이 요구 사항은 kubeconfig로 관리되는 앱 클러스터에 적용됩니다.
  - * kubeconfig 액세스 할 수 있습니다 * : 당신은 액세스 할 수 있습니다 "기본 클러스터 kubecononfig" 그것입니다 "설치하는 동안 를 구성했습니다".
  - * 인증 기관 고려 사항 *: 개인 CA(인증 기관)를 참조하는 kubeconfig 파일을 사용하여 클러스터를 추가하는 경우 에 다음 줄을 추가하십시오 `cluster` kubeconfig 파일의 섹션. 이를 통해 Astra Control이 클러스터를 추가할 수 있습니다.

```
insecure-skip-tls-verify: true
```

  - * Rancher 전용 *: Rancher 환경에서 애플리케이션 클러스터를 관리할 때 Rancher가 제공하는 kubecon무화과 파일에서 애플리케이션 클러스터의 기본 컨텍스트를 수정하여 Rancher API 서버 컨텍스트 대신 컨트롤 플레인 컨텍스트를 사용합니다. 따라서 Rancher API 서버의 부하가 줄어들고 성능이 향상됩니다.
- * Astra Control Provisioner 요구 사항 *: 클러스터를 관리하려면 Astra Trident 구성 요소를 포함하여 올바르게 구성된 Astra Control Provisioner가 있어야 합니다.
  - * Astra Trident 환경 요구 사항 검토 *: Astra Control Provisioner를 설치 또는 업그레이드하기 전에 를 검토하십시오 "지원되는 프런트엔드, 백엔드 및 호스트 구성".
  - * Astra Control Provisioner 기능 활성화 *: Astra Trident 23.10 이상을 설치하고 활성화하는 것이 좋습니다 "Astra Control Provisioner 고급 스토리지 기능". 향후 릴리즈에서 Astra Control Provisioner가 활성화되어 있지 않으면 Astra Control이 Astra Trident를 지원하지 않습니다.
  - * 스토리지 백엔드 구성 *: 최소 하나의 스토리지 백엔드가 있어야 합니다 "Astra Trident에서 구성됨" 클러스터에서.
  - * 스토리지 클래스 구성 *: 최소 하나의 스토리지 클래스가 있어야 합니다 "Astra Trident에서 구성됨" 클러스터에서. 기본 저장소 클래스가 구성된 경우 기본 주석이 있는 * 전용 * 저장소 클래스인지 확인합니다.

- * 볼륨 스냅샷 컨트롤러 구성 및 볼륨 스냅샷 클래스 설치 *: "볼륨 스냅샷 컨트롤러를 설치합니다" 따라서 Astra Control에서 스냅샷을 생성할 수 있습니다. "생성" 하나 이상 `VolumeSnapshotClass` Astra Trident 사용:

자격 검사를 실행합니다

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

단계

1. 실행 중인 Astra Trident 버전을 확인합니다.

```
kubectl get tridentversion -n trident
```

Astra Trident가 있으면 다음과 유사한 출력이 표시됩니다.

```
NAME      VERSION
trident   24.02.0
```

Astra Trident가 없으면 다음과 유사한 출력이 표시됩니다.

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

2. 다음 중 하나를 수행합니다.
   - Astra Trident 23.01 이하를 실행 중인 경우 다음을 사용합니다 "지침" Astra Control Provisioner로 업그레이드하기 전에 Astra Trident의 최신 버전으로 업그레이드하십시오. 가능합니다 "직접 업그레이드를 수행합니다" Astra Trident가 버전 24.02의 4개 릴리즈 윈도우 내에 있는 경우 Astra Control Provisioner 24.02에 등록됩니다. 예를 들어, Astra Trident 23.04에서 Astra Control Provisioner 24.02로 직접 업그레이드할 수 있습니다.
   - Astra Trident 23.10 이상을 실행 중인 경우 Astra Control Provisioner가 설치되었는지 확인합니다 "활성화됨". Astra Control Provisioner는 23.10 이전 Astra Control Center 릴리즈에서 작동하지 않습니다. "Astra Control Provisioner를 업그레이드합니다" 최신 기능에 액세스하기 위해 업그레이드하는 Astra Control Center와 동일한 버전을 사용합니다.
3. 모든 Pod(포함)가 `trident-acp`) 실행 중:

```
kubectl get pods -n trident
```

4. 스토리지 클래스가 지원되는 Astra Trident 드라이버를 사용하고 있는지 확인합니다. 공급자 이름은 이어야 합니다 `csi.trident.netapp.io`. 다음 예를 참조하십시오.

```
kubectl get sc
```

샘플 반응:

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io   Delete          Immediate
true                    5d23h
```

클러스터 역할 **kubecononfig**를 생성합니다

kubeconfig를 사용하여 관리되는 클러스터의 경우 Astra Control Center에 대해 제한된 권한 또는 확장된 권한 관리자 역할을 선택적으로 생성할 수 있습니다. 이미 의 일부로 kubecononfig를 구성했으므로 Astra Control Center 설정에 필요한 절차는 아닙니다 "설치 프로세스".

다음 시나리오 중 하나가 사용자 환경에 적용되는 경우 이 절차를 통해 별도의 kubecononfig를 생성할 수 있습니다.

- 관리하는 클러스터에 대한 Astra Control 권한을 제한하려고 합니다
- 여러 개의 컨텍스트를 사용하며 설치 중에 구성된 기본 Astra Control kubecononfig를 사용할 수 없거나, 단일 컨텍스트의 제한된 역할은 사용자 환경에서 작동하지 않습니다

시작하기 전에

절차 단계를 완료하기 전에 관리하려는 클러스터에 대해 다음 사항을 확인해야 합니다.

- kubbtl v1.23 이상이 설치되었습니다
- Astra Control Center를 통해 추가하고 관리하려는 클러스터에 kubctl 액세스를 허용합니다

> ⓘ 이 절차를 수행하려면 Astra Control Center를 실행 중인 클러스터에 kubectl을 액세스할 필요가 없습니다.

- 활성 컨텍스트에 대한 클러스터 관리자 권한으로 관리하려는 클러스터에 대한 활성 kubecononfig입니다

단계

1. 서비스 계정 생성:
   a. astractrol-service-account.yaML이라는 서비스 계정 파일을 만듭니다.

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

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

   b. 서비스 계정 적용:

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

2. Astra Control에서 클러스터를 관리할 수 있는 충분한 권한을 가진 다음 클러스터 역할 중 하나를 생성합니다.

제한된 클러스터 역할

이 역할에는 Astra Control에서 관리할 클러스터를 관리하는 데 필요한 최소 권한이 포함되어 있습니다.

a. 을 생성합니다 `ClusterRole` 호출되는 파일(예: `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
  - horizontalpodautoscalers
  - ingresses
  - jobs
  - namespaces
  - networkpolicies
  - persistentvolumeclaims
  - poddisruptionbudgets
  - pods
  - podtemplates
  - replicasets
  - replicationcontrollers
  - replicationcontrollers/scale
  - rolebindings
  - roles
  - secrets
  - serviceaccounts
  - services
  - statefulsets
  - tridentmirrorrelationships
  - tridentsnapshotinfos
  - volumesnapshots
  - volumesnapshotcontents
  verbs:
  - delete

# Watch resources
# Necessary to monitor progress
- apiGroups:
  - ""
  resources:
  - pods
  - replicationcontrollers
  - replicationcontrollers/scale
  verbs:
  - watch

# Update resources
- apiGroups:
  - ""
  - build.openshift.io
  - image.openshift.io
  resources:
  - builds/details
  - replicationcontrollers
  - replicationcontrollers/scale
  - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

b. (OpenShift 클러스터에만 해당) 의 끝에 다음을 추가합니다 `astra-admin-account.yaml` 파일:

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

c. 클러스터 역할 적용:

```
kubectl apply -f astra-admin-account.yaml
```

클러스터 역할이 확장되었습니다

이 역할에는 Astra Control에서 관리할 클러스터에 대한 확장된 권한이 포함됩니다. 여러 컨텍스트를 사용하고 설치 중에 구성된 기본 Astra Control kubecononfig를 사용할 수 없거나 단일 컨텍스트의 제한된 역할을 사용할 수 없는 경우 이 역할을 사용할 수 있습니다.

다음 사항을 참조하십시오 `ClusterRole` 일반 Kubernetes의 예는 단계입니다. 사용자 환경에 대한 지침은 Kubernetes 배포 문서를 참조하십시오.

a. 을 생성합니다 `ClusterRole` 호출되는 파일(예: `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. 클러스터 역할 적용:

```
kubectl apply -f astra-admin-account.yaml
```

3. 클러스터 역할에 대한 클러스터 역할 바인딩을 서비스 계정에 생성합니다.

a. astracontrol-clusterrobinding.YAML이라는 ClusterRoleBinding 파일을 만듭니다.

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

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

b. 클러스터 역할 바인딩을 적용합니다.

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

4. 토큰 암호 생성 및 적용:
   a. 라는 토큰 비밀 파일을 만듭니다 `secret-astracontrol-service-account.yaml`.

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

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. 토큰 암호 적용:

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

5. 토큰 암호를 에 추가하여 서비스 계정에 추가합니다 `secrets` 배열(다음 예제의 마지막 줄):

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. '<context>'을(를) 설치에 적합한 컨텍스트로 대체하여 서비스 계정 암호를 나열합니다.

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

출력의 끝은 다음과 유사합니다.

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

의 각 요소에 대한 인덱스입니다 `secrets` 어레이는 0으로 시작합니다. 위의 예에서 의 인덱스입니다 `astracontrol-service-account-dockercfg-48xhx` 는 0이고 의 인덱스입니다 `secret-astracontrol-service-account` 1입니다. 출력에서 서비스 계정의 인덱스 번호를 기록해 둡니다. 다음 단계에서는 이 인덱스 번호가 필요합니다.

7. 다음과 같이 kubecononfig를 생성합니다.
   a. 을 생성합니다 `create-kubeconfig.sh` 파일.
   b. 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
```

c. Kubernetes 클러스터에 적용할 명령을 소스 하십시오.

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

8. (선택 사항) kubeconfig의 이름을 클러스터의 의미 있는 이름으로 바꿉니다.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(기술 미리 보기) 관리형 클러스터를 위한 **Astra Connector**를 설치합니다

Astra Control Center에서 관리되는 클러스터는 Astra Connector를 사용하여 관리형 클러스터와 Astra Control Center 간의 통신을 지원합니다. 관리하려는 모든 클러스터에 Astra Connector를 설치해야 합니다.

Astra Connector를 설치합니다

Kubernetes 명령 및 CR(Custom Resource) 파일을 사용하여 Astra Connector를 설치합니다.

이 작업에 대해

- 이러한 단계를 수행할 때 Astra Control으로 관리하려는 클러스터에서 다음 명령을 실행합니다.
- 방호 호스트를 사용하는 경우 방호 호스트의 명령줄에서 이러한 명령을 실행하십시오.

시작하기 전에

- Astra Control을 사용하여 관리할 클러스터에 액세스해야 합니다.
- 클러스터에 Astra Connector 연산자를 설치하려면 Kubernetes 관리자 권한이 필요합니다.

Kubernetes 1.25 이상 클러스터의 기본값인 Pod 보안 승인 적용을 사용하여 클러스터를 구성한 경우 해당 네임스페이스에 PSA 제한을 활성화해야 합니다. 을 참조하십시오 "Astra Control을 사용하여 클러스터 관리를 위한 환경을 준비합니다" 를 참조하십시오.

단계

1. Astra Control으로 관리할 클러스터에 Astra Connector 연산자를 설치합니다. 이 명령을 실행하면 네임스페이스가 생성됩니다 `astra-connector-operator` 이 만들어지고 구성이 네임스페이스에 적용됩니다.

```
kubectl apply -f https://github.com/NetApp/astra-connector-
operator/releases/download/24.02.0-
202403151353/astraconnector_operator.yaml
```

2. 작업자가 설치되어 있고 준비가 되었는지 확인합니다.

```
kubectl get all -n astra-connector-operator
```

3. Astra Control에서 API 토큰을 받습니다. 을 참조하십시오 "Astra 자동화 문서" 를 참조하십시오.
4. 토큰을 사용하여 암호를 생성합니다. <API_TOKEN>를 Astra Control에서 받은 토큰으로 대체:

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. Astra Connector 이미지를 가져오는 데 사용할 Docker 암호를 생성합니다. 괄호 안의 값을 사용자 환경의 정보로 대체:

<ASTRA_CONTROL_ACCOUNT_ID>는 Astra Control 웹 UI에서 찾을 수 있습니다. 웹 UI에서 페이지 오른쪽 상단의 그림 아이콘을 선택하고 * API access * 를 선택합니다.

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. Astra Connector CR 파일을 생성하고 이름을 지정합니다 `astra-connector-cr.yaml`. 괄호 <> 의 값을 Astra Control 환경 및 클러스터 구성과 일치하도록 업데이트합니다.
   - <ASTRA_CONTROL_ACCOUNT_ID>: 이전 단계에서 Astra Control 웹 UI에서 구함.
   - <CLUSTER_NAME>: Astra Control에서 이 클러스터를 할당해야 하는 이름입니다.
   - <ASTRA_CONTROL_URL>: Astra Control의 웹 UI URL입니다. 예를 들면 다음과 같습니다.

```
https://astra.control.url
```

```
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. 를 채운 후 `astra-connector-cr.yaml` 올바른 값이 있는 파일에 CR을 적용합니다.

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. Astra Connector가 완전히 구축되었는지 확인:

```
kubectl get all -n astra-connector
```

9. 클러스터가 Astra Control에 등록되었는지 확인:

```
kubectl get astraconnectors.astra.netapp.io -A
```

다음과 유사한 출력이 표시됩니다.

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e   Registered with Astra
```

10. Astra Control 웹 UI의 * 클러스터 * 페이지에서 관리되는 클러스터 목록에 클러스터가 나타나는지 확인합니다.

클러스터를 추가합니다

앱 관리를 시작하려면 Kubernetes 클러스터를 추가하고 이를 컴퓨팅 리소스로 관리합니다. Kubernetes 애플리케이션을 검색하려면 Astra Control Center용 클러스터를 추가해야 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.

시작하기 전에

- 클러스터를 추가하기 전에 필요한 를 검토 및 수행합니다 "선행 작업".
- ONTAP SAN 드라이버를 사용하는 경우 모든 Kubernetes 클러스터에서 다중 경로가 활성화되어 있는지 확인하십시오.

단계

1. 대시보드 또는 클러스터 메뉴에서 이동합니다.
   - 리소스 요약의 * 대시보드 * 에서 클러스터 창에서 * 추가 * 를 선택합니다.
   - 왼쪽 탐색 영역에서 * 클러스터 * 를 선택한 다음 클러스터 페이지에서 * 클러스터 추가 * 를 선택합니다.
2. Add Cluster * (클러스터 추가 *) 창이 열리면 kubecononfig.YAML 파일을 업로드하거나 kubecononfig.YAML 파일의 내용을 붙여 넣습니다.

   "kubecononfig.yAML" 파일에는 하나의 클러스터에 대한 클러스터 자격 증명만 * 포함되어야 합니다.

   직접 만드는 경우 `kubeconfig` 파일에서 * 하나의 * 컨텍스트 요소만 정의해야 합니다. 을 참조하십시오 "Kubernetes 문서" 을 참조하십시오 `kubeconfig` 파일. 을 사용하여 제한된 클러스터 역할에 대해 kubecon무화과를 생성한 경우 "알려 드립니다"이 단계에서는 과베토화과를 업로드하거나 붙여 넣으십시오.

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

   Astra Control Provisioner에서 구성되고 ONTAP 스토리지에서 지원하는 스토리지 클래스를 선택해야 합니다.

6. 정보를 검토하고 모든 것이 정상적으로 나타나면 * 추가 * 를 선택합니다.

결과

클러스터가 * 검색 * 상태로 전환되고 * 정상 * 으로 변경됩니다. 이제 Astra Control Center로 클러스터를 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.

ONTAP 스토리지 백엔드에서 인증을 설정합니다

Astra Control Center는 ONTAP 백엔드를 인증하는 두 가지 모드를 제공합니다.

- * 자격 증명 기반 인증 *: 필요한 권한이 있는 ONTAP 사용자의 사용자 이름 및 암호입니다. ONTAP 버전과의 호환성을 최대화하려면 admin 또는 vsadmin과 같이 미리 정의된 보안 로그인 역할을 사용해야 합니다.
- * 인증서 기반 인증 *: Astra Control Center는 백엔드에 설치된 인증서를 사용하여 ONTAP 클러스터와 통신할 수도 있습니다. 클라이언트 인증서, 키 및 신뢰할 수 있는 CA 인증서를 사용해야 합니다(권장).

나중에 기존 백엔드를 업데이트하여 한 가지 인증 유형에서 다른 방법으로 이동할 수 있습니다. 한 번에 하나의 인증 방법만 지원됩니다.

자격 증명 기반 인증을 사용합니다

Astra Control Center에는 클러스터 범위에 대한 자격 증명이 필요합니다 `admin` ONTAP 백엔드와 통신합니다. 과 같이 미리 정의된 표준 역할을 사용해야 합니다 `admin`. 이를 통해 향후 Astra Control Center 릴리스에서 사용할 기능 API를 노출할 수 있는 향후 ONTAP 릴리스와 향후 호환될 수 있습니다.

> 사용자 지정 보안 로그인 역할은 Astra Control Center에서 생성 및 사용할 수 있지만 권장되지 않습니다.

백엔드 정의의 예는 다음과 같습니다.

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

백엔드 정의만 자격 증명이 일반 텍스트로 저장되는 곳입니다. 백엔드의 생성 또는 업데이트는 자격 증명에 대한 지식이 필요한 유일한 단계입니다. 따라서 Kubernetes 또는 스토리지 관리자가 수행할 수 있는 관리자 전용 작업입니다.

Astra Control Center는 인증서를 사용하여 신규 및 기존 ONTAP 백엔드와 통신할 수 있습니다. 백엔드 정의에 다음 정보를 입력해야 합니다.

- `clientCertificate`: 클라이언트 인증서.
- `clientPrivateKey`: 연결된 개인 키.
- `trustedCACertificate`: 신뢰할 수 있는 CA 인증서입니다. 신뢰할 수 있는 CA를 사용하는 경우 이 매개 변수를 제공해야 합니다. 신뢰할 수 있는 CA가 사용되지 않으면 이 작업을 무시할 수 있습니다.

다음 유형의 인증서 중 하나를 사용할 수 있습니다.

- 자체 서명된 인증서
- 타사 인증서입니다

자체 서명된 인증서를 사용하여 인증을 활성화합니다

일반적인 워크플로에는 다음 단계가 포함됩니다.

단계

1. 클라이언트 인증서 및 키를 생성합니다. 생성 시 CN(일반 이름)을 ONTAP 사용자로 설정하여 인증하십시오.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. 유형의 클라이언트 인증서를 설치합니다 `client-ca` ONTAP 클러스터의 키입니다.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. ONTAP 보안 로그인 역할이 인증서 인증 방법을 지원하는지 확인합니다.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. 생성된 인증서를 사용하여 인증을 테스트합니다. ONTAP 관리 LIF> 및 <vserver name>를 관리 LIF IP 및 SVM 이름으로 바꿉니다. LIF의 서비스 정책이 으로 설정되어 있는지 확인해야 합니다 `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. 이전 단계에서 얻은 값을 사용하여 Astra Control Center UI에 스토리지 백엔드를 추가합니다.

타사 인증서로 인증을 활성화합니다

타사 인증서가 있는 경우 다음 단계를 사용하여 인증서 기반 인증을 설정할 수 있습니다.

단계

1. 개인 키와 CSR을 생성합니다.

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. CSR을 Windows CA(타사 CA)로 전달하고 서명된 인증서를 발급합니다.
3. 서명된 인증서를 다운로드하고 이름을 'ONTAP_signed_cert.crt’로 지정합니다.
4. Windows CA(타사 CA)에서 루트 인증서를 내보냅니다.
5. 이 파일의 이름을 지정합니다 `ca_root.crt`

   이제 다음 세 개의 파일이 있습니다.

   - * 개인 키 *: `ontap_signed_request.key` (이 키는 ONTAP의 서버 인증서에 해당하는 키입니다. 서버 인증서를 설치하는 동안 필요합니다.)
   - * 서명된 인증서 *: `ontap_signed_cert.crt` (ONTAP에서 _server certificate_라고도 함)
   - * 루트 CA 인증서 *: `ca_root.crt` (ONTAP에서 _server-ca certificate_라고도 합니다.)
6. 이러한 인증서를 ONTAP에 설치합니다. 생성 및 설치 `server` 및 `server-ca` ONTAP의 인증서.

YAML의 샘플을 확장합니다

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. 암호 없는 통신을 위해 동일한 호스트에 대한 클라이언트 인증서를 생성합니다. Astra Control Center는 이 프로세스를 사용하여 ONTAP와 통신합니다.
8. ONTAP에서 클라이언트 인증서 생성 및 설치:

YAML의 샘플을 확장합니다

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate's CA and serial number for reference:

CA:
serial:
The certificate's generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```


9. Astra Control Center UI에 스토리지 백엔드를 추가하고 다음 값을 제공합니다.
   - * 클라이언트 인증서 *: ONTAP_TEST_CLIENT.PEM
   - * 개인 키 *: ontap_test_client.key
   - * 신뢰할 수 있는 CA 인증서 *: ONTAP_signed_certt. CRT

스토리지 백엔드를 추가합니다

자격 증명 또는 인증서 인증 정보를 설정한 후 기존 ONTAP 스토리지 백엔드를 Astra Control Center에 추가하여 리소스를 관리할 수 있습니다.

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

Astra Control Provisioner를 사용하도록 설정한 경우 NetApp SnapMirror 기술을 사용할 때 Astra Control Center에서 ONTAP 스토리지 백엔드를 추가 및 관리하는 것은 선택 사항입니다.

단계

1. 왼쪽 탐색 영역의 대시보드에서 * backends * 를 선택합니다.
2. 추가 * 를 선택합니다.
3. 스토리지 백엔드 추가 페이지의 기존 사용 섹션에서 * ONTAP * 를 선택합니다.
4. 다음 중 하나를 선택합니다.
   - * 관리자 자격 증명 사용 *: ONTAP 클러스터 관리 IP 주소와 관리 자격 증명을 입력합니다. 자격 증명은 클러스터 전체의 자격 증명이어야 합니다.

     여기에 자격 증명을 입력한 사용자에게는 가 있어야 합니다 `ontapi` ONTAP 클러스터의 ONTAP System Manager에서 활성화된 사용자 로그인 액세스 방법입니다. SnapMirror 복제를 사용하려는 경우 액세스 방법이 있는 "admin" 역할의 사용자 자격 증명을 적용하십시오 `ontapi` 및 `http`, 소스 및 대상 ONTAP 클러스터 모두에서. 을 참조하십시오 "ONTAP 설명서에서 사용자 계정을 관리합니다" 를 참조하십시오.
   - * 인증서 사용 *: 인증서를 업로드합니다 `.pem` 파일, 인증서 키입니다 `.key` 파일 및 인증 기관 파일(옵션)을 선택합니다.
5. 다음 * 을 선택합니다.
6. 백엔드 세부 정보를 확인하고 * 관리 * 를 선택합니다.

결과

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

백엔드가 표시되도록 페이지를 새로 고쳐야 할 수 있습니다.

버킷을 추가합니다

Astra Control UI 또는 를 사용하여 버킷을 추가할 수 있습니다 "Astra Control API를 참조하십시오". 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.

애플리케이션 구성과 영구 스토리지를 동일한 클러스터에 클론 복제하려는 경우 Astra Control에 버킷이 필요하지 않습니다. 애플리케이션 스냅샷 기능에는 버킷이 필요하지 않습니다.

시작하기 전에

- Astra Control Center에서 관리하는 클러스터에서 연결할 수 있는 버킷이 있어야 합니다.
- 버킷에 대한 자격 증명이 있는지 확인하십시오.
- 버킷이 다음 유형 중 하나인지 확인합니다.
  - NetApp ONTAP S3
  - NetApp StorageGRID S3
  - Microsoft Azure를 참조하십시오
  - 일반 S3

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

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.

단계

1. 왼쪽 탐색 영역에서 * Bucket * 을 선택합니다.
2. 추가 * 를 선택합니다.
3. 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.

4. 기존 버킷 이름과 선택적 설명을 입력합니다.

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

5. S3 엔드포인트의 이름 또는 IP 주소를 입력합니다.
6. 자격 증명 선택 * 에서 * 추가 * 또는 * 기존 * 사용 탭을 선택합니다.
   - 추가 * 를 선택한 경우:
     i. Astra Control의 다른 자격 증명과 구별되는 자격 증명의 이름을 입력합니다.
     ii. 클립보드의 내용을 붙여 넣어 액세스 ID와 비밀 키를 입력합니다.
   - 기존 사용 * 을 선택한 경우:
     i. 버킷에 사용할 기존 자격 증명을 선택합니다.
7. 를 선택합니다 `Add`.

   버킷을 추가하면 Astra Control이 기본 버킷 표시기로 하나의 버킷을 표시합니다. 사용자가 만든 첫 번째 버킷이 기본 버킷이 됩니다. 양동이 추가될 때 나중에 결정할 수 있습니다 "다른 기본 버킷을 설정합니다".