설치 개요

Astra Control Center

NetApp
November 21, 2023

목차

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 NetApp Support 사이트에서 설치 번들을 다운로드하고 다음 단계를 수행하십시오. 이 절차를 사용하여 인터넷에 연결되었거나 공기가 연결된 환경에 Astra Control Center를 설치할 수 있습니다.

기타 설치 절차

- RedHat OpenShift OperatorHub * 로 설치: 이 옵션을 사용합니다 "대체 절차" OperatorHub를 사용하여 OpenShift에 Astra Control Center를 설치합니다.
- * 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.

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

시작하기 전에

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- 사용자 환경에서 POD 보안 정책을 구성했거나 구성하려는 경우 POD 보안 정책 및 해당 정책이 Astra Control Center 설치에 어떤 영향을 미치는지 숙지하십시오. 을 참조하십시오 "POD 보안 정책 제한 사항 이해".
- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

```
kubectl get apiservices
```

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

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

이 작업에 대해

Astra Control Center 설치 프로세스를 통해 다음을 수행할 수 있습니다.

- 에 Astra 구성 요소를 설치합니다 `netapp-acc` (또는 사용자 지정 이름) 네임스페이스입니다.
- 기본 Astra Control Owner 관리자 계정을 생성합니다.
- 관리자 이메일 주소와 기본 초기 설정 암호를 설정합니다. 이 사용자에게는 UI에 처음 로그인하는 데 필요한 소유자 역할이 할당됩니다.
- 모든 Astra Control Center Pod가 실행되고 있는지 확인합니다.
- Astra Control Center UI를 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.

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center를 다운로드하고 압축을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다
- Astra Control Center 운영자를 설치합니다
- Astra Control Center를 구성합니다
- Astra 제어 센터 및 운전자 설치를 완료합니다
- 시스템 상태를 확인합니다
- 부하 분산을 위한 수신 설정
- Astra Control Center UI에 로그인합니다

Astra Control Center를 다운로드하고 압축을 풉니다

1. 로 이동합니다 "Astra Control Center 다운로드 페이지" 를 방문하십시오.
2. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 클릭합니다.
3. (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.

4. Astra Control Center 번들에서 이미지를 추출합니다.

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

NetApp Astra kubtl 플러그인을 설치합니다

NetApp Astra kubtl 명령줄 플러그인을 사용하여 이미지를 로컬 Docker 저장소로 푸시할 수 있습니다.

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
```

이미지를 로컬 레지스트리에 추가합니다

1. 용기 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

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

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 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

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

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 레지스트리에 로그인합니다.

   ```
   podman login <YOUR_REGISTRY>
   ```

3. 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=23.04.2-7
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=23.04.2-7
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://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. Astra Control Center 호스트 클러스터에 대한 KUBECONFIG를 내보냅니다.

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

설치를 완료하기 전에 KUBECONFIG가 Astra Control Center를 설치할 클러스터를 가리키고 있는지 확인하십시오. KUBECONFIG는 하나의 컨텍스트만 포함할 수 있습니다.

2. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

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

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

   응답:

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

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

   자리 표시자입니다 `your_registry_path` 이전에 업로드한 이미지의 위치와 일치해야 합니다(예: `[Registry_URL]/netapp/astra/astracc/23.04.2-7`)를 클릭합니다.

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

   샘플 반응:

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

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

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

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

   샘플 반응:

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

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

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

응답

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

Astra Control Center 운영자를 설치합니다

1. 디렉토리를 변경합니다.

```
cd manifests
```

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

```
vim astra_control_center_operator_deploy.yaml
```

YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

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

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

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

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

```
<strong>astra_control_center_operator_deploy.yaml</strong>
```

```
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: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: [your_registry_path]/acc-operator:23.04.36
        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
```

3. Astra Control Center 운영자를 설치합니다.

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

샘플 반응:

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

4. Pod가 실행 중인지 확인합니다.

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

Astra Control 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.

<code>accountName</code>

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

<code>astraVersion</code>

설정	지침	유형	예
`astraVersion`	배포할 Astra Control Center의 버전입니다. 값이 미리 채워질 수 있으므로 이 설정에 대한 작업은 필요하지 않습니다.	문자열	`23.04.2-7`

<code>astraAddress</code>

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`

<code>autoSupport</code>

이 섹션에서 어떤 항목을 선택하는지에 따라 NetApp의 사전 지원 애플리케이션인 NetApp Active IQ에 참여할 것인지, 그리고 데이터를 보낼 위치를 결정할 수 있습니다. 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 보낼 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`

<code>email</code>

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`

<code>firstName</code>

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

<code>LastName</code>

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

<code>imageRegistry</code>

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	에서 이미지를 푸시한 이미지 레지스트리의 이름입니다 이전 단계. 사용하지 마십시오 `http://` 또는 `https://` 레지스트리 이름.	문자열	`example.registry.com/astra`
`imageRegistry.secret`	에 대해 입력한 문자열인 경우 필수입니다 `imageRegistry.name' requires a secret.` `IMPORTANT: If you are using a registry that does not require authorization, you must delete this` `secret` 줄 내부 `imageRegistry` 그렇지 않으면 설치가 실패합니다.	이미지 레지스트리를 인증하는 데 사용되는 Kubernetes 비밀의 이름입니다.	문자열	`astra-registry-cred`

<code>storageClass</code>

설정	지침	유형	예
`storageClass`	를 변경합니다 `storageClass` 값 시작 `ontap-gold` 을 다른 Astra Trident storageClass 리소스로 이동합니다. 명령을 실행합니다 `kubectl get sc` 구성된 기존 스토리지 클래스를 확인하려면 다음을 수행합니다. 매니페스트 파일에 Astra Trident 기반 스토리지 클래스 중 하나를 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`

<code>volumeReclaimPolicy</code>

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`

<code>ingressType</code>

설정	지침	유형	옵션
`ingressType`	다음 수신 유형 중 하나를 사용하십시오. **`Generic`** (`ingressType: "Generic"`) (기본값) 다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 배포한 후 을 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. 참고: "로드 밸런서" 및 수신 서비스 유형에 대한 자세한 내용은 을	문자열	• `Generic` (기본값) • `AccTraefik`

<code>scaleSize</code>

설정	지침	유형	옵션
`scaleSize`	기본적으로 Astra는 HA(High Availability)를 사용합니다. `scaleSize`의 `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)

<code>astraResourcesScaler</code>

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`

<code>additionalValues</code>

◦ Astral Control Center 및 Cloud Insights 통신의 경우 TLS 인증서 확인은 기본적으로 비활성화되어 있습니다. 에 다음 섹션을 추가하여 Cloud Insights와 Astra Control Center 호스트 클러스터 및 관리 클러스터 간의 통신에 대한 TLS 인증 검증을 활성화할 수 있습니다 `additionalValues`.

```
additionalValues:
  netapp-monitoring-operator:
    config:
      ciSkipTlsVerify: false
  cloud-insights-service:
    config:
      ciSkipTlsVerify: false
  telemetry-service:
    config:
      ciSkipTlsVerify: false
```

<code>crds</code>

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.

```
<strong>astra_control_center.yaml</strong>
```

```
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: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues: {}
  crds:
    externalTraefik: false
    externalCertManager: false
```

Astra 제어 센터 및 운전자 설치를 완료합니다

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

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

   샘플 반응:

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

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

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

   샘플 반응:

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

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

샘플 응답

```
NAME                                          READY   STATUS
RESTARTS     AGE
acc-helm-repo-6cc7696d8f-pmhm8                1/1     Running     0
9h
activity-597fb656dc-5rd4l                     1/1     Running     0
9h
activity-597fb656dc-mqmcw                     1/1     Running     0
9h
api-token-authentication-62f84                1/1     Running     0
9h
api-token-authentication-68nlf                1/1     Running     0
9h
api-token-authentication-ztgrm                1/1     Running     0
9h
asup-669d4ddbc4-fnmwp                         1/1     Running     1
(9h ago)   9h
authentication-78789d7549-lk686               1/1     Running     0
9h
bucketservice-65c7d95496-24x7l                1/1     Running     3
(9h ago)   9h
cert-manager-c9f9fbf9f-k8zq2                  1/1     Running     0
9h
cert-manager-c9f9fbf9f-qjlzm                  1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-b5qll       1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-p5whs       1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-4722b         1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-86kv5         1/1     Running     0
9h
certificates-59d9f6f4bd-2j899                 1/1     Running     0
9h
certificates-59d9f6f4bd-9d9k6                 1/1     Running     0
9h
certificates-expiry-check-28011180--1-8lkxz   0/1     Completed   0
9h
cloud-extension-5c9c9958f8-jdhrp              1/1     Running     0
9h
cloud-insights-service-5cdd5f7f-pp8r5         1/1     Running     0
9h
composite-compute-66585789f4-hxn5w            1/1     Running     0
```

```
9h
composite-volume-68649f68fd-tb7p4             1/1     Running     0
9h
credentials-dfc844c57-jsx92                   1/1     Running     0
9h
credentials-dfc844c57-xw26s                   1/1     Running     0
9h
entitlement-7b47769b87-4jb6c                  1/1     Running     0
9h
features-854d8444cc-c24b7                     1/1     Running     0
9h
features-854d8444cc-dv6sm                     1/1     Running     0
9h
fluent-bit-ds-9tlv4                           1/1     Running     0
9h
fluent-bit-ds-bpkcb                           1/1     Running     0
9h
fluent-bit-ds-cxmwx                           1/1     Running     0
9h
fluent-bit-ds-jgnhc                           1/1     Running     0
9h
fluent-bit-ds-vtr6k                           1/1     Running     0
9h
fluent-bit-ds-vxqd5                           1/1     Running     0
9h
graphql-server-7d4b9d44d5-zdbf5               1/1     Running     0
9h
identity-6655c48769-4pwk8                     1/1     Running     0
9h
influxdb2-0                                   1/1     Running     0
9h
keycloak-operator-55479d6fc6-slvmt            1/1     Running     0
9h
krakend-f487cb465-78679                       1/1     Running     0
9h
krakend-f487cb465-rjsxx                       1/1     Running     0
9h
license-64cbc7cd9c-qxsr8                      1/1     Running     0
9h
login-ui-5db89b5589-ndb96                     1/1     Running     0
9h
loki-0                                        1/1     Running     0
9h
metrics-facade-8446f64c94-x8h7b               1/1     Running     0
9h
monitoring-operator-6b44586965-pvcl4          2/2     Running     0
```

```
9h
nats-0                                            1/1     Running     0
9h
nats-1                                            1/1     Running     0
9h
nats-2                                            1/1     Running     0
9h
nautilus-85754d87d7-756qb                         1/1     Running     0
9h
nautilus-85754d87d7-q8j7d                         1/1     Running     0
9h
openapi-5f9cc76544-7fnjm                          1/1     Running     0
9h
openapi-5f9cc76544-vzr7b                          1/1     Running     0
9h
packages-5db49f8b5-lrzhd                          1/1     Running     0
9h
polaris-consul-consul-server-0                    1/1     Running     0
9h
polaris-consul-consul-server-1                    1/1     Running     0
9h
polaris-consul-consul-server-2                    1/1     Running     0
9h
polaris-keycloak-0                                1/1     Running     2
(9h ago)   9h
polaris-keycloak-1                                1/1     Running     0
9h
polaris-keycloak-2                                1/1     Running     0
9h
polaris-keycloak-db-0                             1/1     Running     0
9h
polaris-keycloak-db-1                             1/1     Running     0
9h
polaris-keycloak-db-2                             1/1     Running     0
9h
polaris-mongodb-0                                 1/1     Running     0
9h
polaris-mongodb-1                                 1/1     Running     0
9h
polaris-mongodb-2                                 1/1     Running     0
9h
polaris-ui-66fb99479-qp9gq                        1/1     Running     0
9h
polaris-vault-0                                   1/1     Running     0
9h
polaris-vault-1                                   1/1     Running     0
```

```
9h
polaris-vault-2                                   1/1     Running     0
9h
public-metrics-76fbf9594d-zmxzw                   1/1     Running     0
9h
storage-backend-metrics-7d7fbc9cb9-lmd25          1/1     Running     0
9h
storage-provider-5bdd456c4b-2fftc                 1/1     Running     0
9h
task-service-87575df85-dnn2q                      1/1     Running     3
(9h ago)   9h
task-service-task-purge-28011720--1-q6w4r         0/1     Completed   0
28m
task-service-task-purge-28011735--1-vk6pd         1/1     Running     0
13m
telegraf-ds-2r2kw                                 1/1     Running     0
9h
telegraf-ds-6s9d5                                 1/1     Running     0
9h
telegraf-ds-96jl7                                 1/1     Running     0
9h
telegraf-ds-hbp84                                 1/1     Running     0
9h
telegraf-ds-plwzv                                 1/1     Running     0
9h
telegraf-ds-sr22c                                 1/1     Running     0
9h
telegraf-rs-4sbg8                                 1/1     Running     0
9h
telemetry-service-fb9559f7b-mk9l7                 1/1     Running     3
(9h ago)   9h
tenancy-559bbc6b48-5msgg                          1/1     Running     0
9h
traefik-d997b8877-7xpf4                           1/1     Running     0
9h
traefik-d997b8877-9xv96                           1/1     Running     0
9h
trident-svc-585c97548c-d25z5                      1/1     Running     0
9h
vault-controller-88484b454-2d6sr                  1/1     Running     0
9h
vault-controller-88484b454-fc5cz                  1/1     Running     0
9h
vault-controller-88484b454-jktld                  1/1     Running     0
9h
```

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

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

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

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

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

응답:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  23.04.2-7   10.111.111.111
True
```

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

부하 분산을 위한 수신 설정

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 Center를 배포한 후 URL을 사용하여 Astra Control Center를 노출하도록 수신 컨트롤러를 구성해야 합니다.

설치 단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다. Astra Control Center는 다양한 수신 컨트롤러 유형을 지원합니다. 이러한 설정 절차에서는 다음과 같은 수신 컨트롤러 유형에 대한 예제 단계를 제공합니다.

- 이스티오 침투
- Nginx 수신 컨트롤러
- OpenShift 수신 컨트롤러

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

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

에 서비스가 있는 경우 `Error` 상태, 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
```

다음 단계

- (선택 사항) 환경에 따라 사후 설치를 완료합니다 "구성 단계".
- 를 수행하여 배포를 완료합니다 "설정 작업".

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 전제 조건 충족 *: "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- * 정상적인 클러스터 운영자 및 API 서비스 *:
  - OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다.

    ```
    oc get clusteroperators
    ```

  - OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다.

    ```
    oc get apiservices
    ```

- * FQDN 주소 *: 데이터 센터의 Astra Control Center에 대한 FQDN 주소를 얻습니다.
- * OpenShift Permissions *: Red Hat OpenShift Container Platform에 대한 필수 권한과 액세스를 얻어 설명된 설치 단계를 수행합니다.
- 인증서 관리자 구성됨 *: 인증서 관리자가 클러스터에 이미 있는 경우 일부를 수행해야 합니다 "필수 단계" 따라서 Astra Control Center는 자체 인증 관리자를 설치하지 않습니다. 기본적으로 Astra Control Center는 설치 중에 자체 인증서 관리자를 설치합니다.
- * Kubernetes 수신 컨트롤러 *: 클러스터의 로드 밸런싱과 같은 서비스에 대한 외부 액세스를 관리하는 Kubernetes 수신 컨트롤러가 있는 경우 Astra Control Center와 함께 사용하도록 설정해야 합니다.
  a. 연산자 네임스페이스 만들기:

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

  b. "설정을 완료합니다" 수신 컨트롤러 유형에 적합합니다.

단계

- Astra Control Center를 다운로드하고 압축을 풉니다
- NetApp Astra kubtl 플러그인을 설치합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 운영자 설치 페이지를 찾으십시오

- 운전자를 설치합니다
- Astra Control Center를 설치합니다

Astra Control Center를 다운로드하고 압축을 풉니다

1. 로 이동합니다 "Astra Control Center 다운로드 페이지" 를 방문하십시오.
2. Astra Control Center가 포함된 번들을 다운로드합니다 (`astra-control-center-[version].tar.gz`)를 클릭합니다.
3. (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.

4. Astra Control Center 번들에서 이미지를 추출합니다.

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

NetApp Astra kubtl 플러그인을 설치합니다

NetApp Astra kubtl 명령줄 플러그인을 사용하여 이미지를 로컬 Docker 저장소로 푸시할 수 있습니다.

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.

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
```

이미지를 로컬 레지스트리에 추가합니다

1. 용기 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

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

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 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

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

   ```
   acc.manifest.bundle.yaml
   acc/
   ```

2. 레지스트리에 로그인합니다.

   ```
   podman login <YOUR_REGISTRY>
   ```

3. 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=23.04.2-7
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=23.04.2-7
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://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

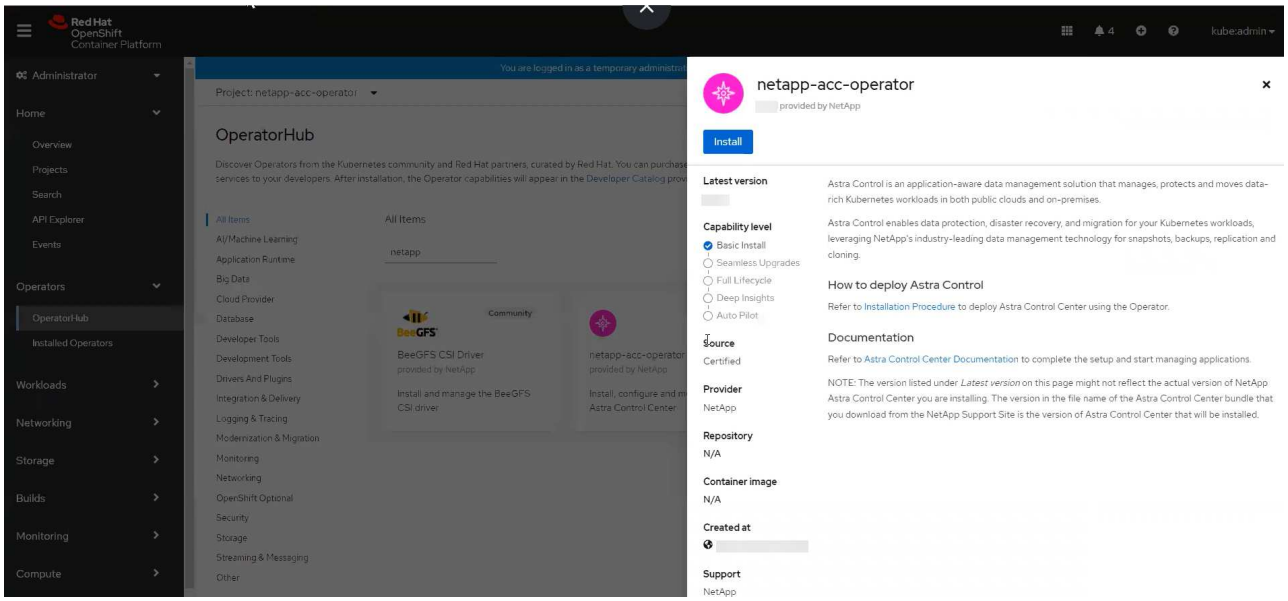
운영자 설치 페이지를 찾으십시오

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

i. OpenShift Container Platform UI에 로그인합니다.

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

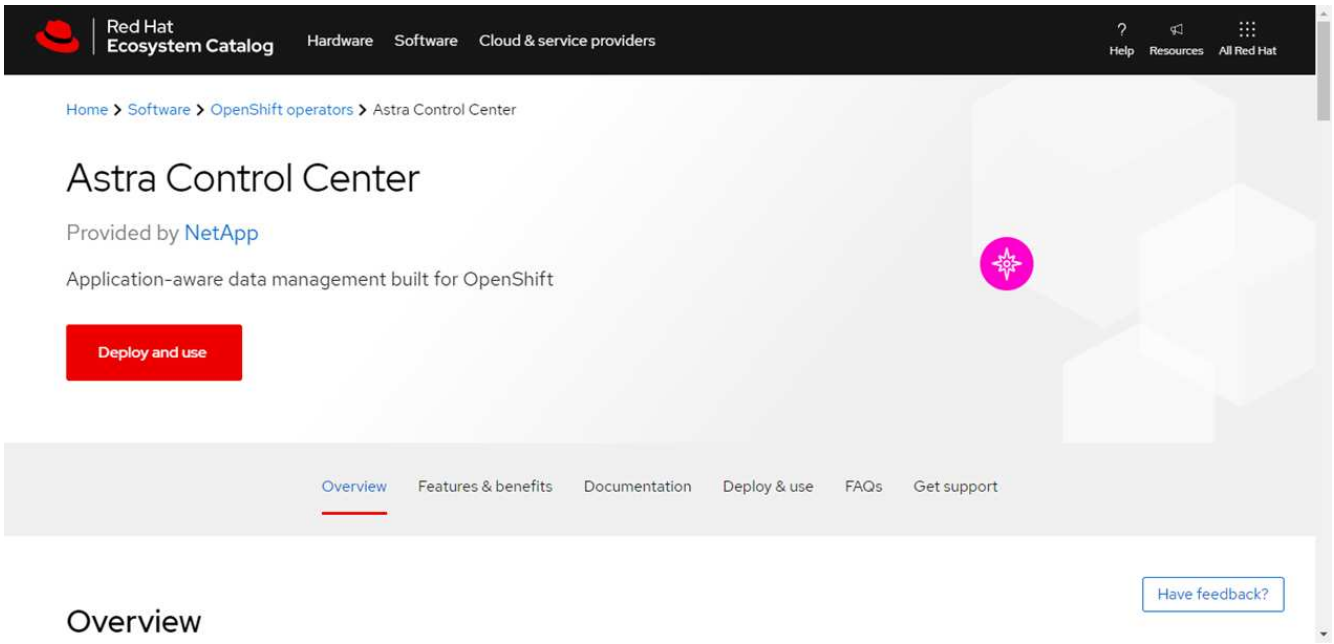
iii. NetApp Astra Control Center 운영자를 검색하여 선택합니다.



◦ Red Hat 에코시스템 카탈로그:

i. NetApp Astra Control Center를 선택합니다 "운영자".

ii. 배포 및 사용 * 을 선택합니다.



운전자를 설치합니다

1. Install Operator * 페이지를 완료하고 운영자를 설치합니다.

운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

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

b. 수동 또는 자동 승인 전략을 선택합니다.

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

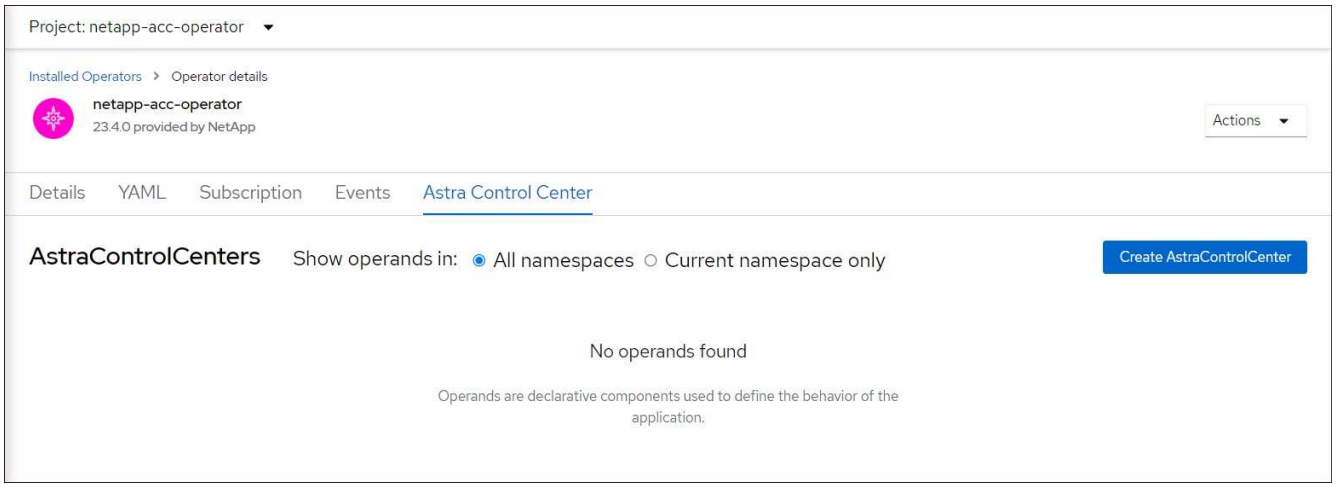
c. 설치 * 를 선택합니다.

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

2. 콘솔에서 OperatorHub 메뉴로 이동하여 운영자가 성공적으로 설치되었는지 확인합니다.

Astra Control Center를 설치합니다

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



2. 를 완료합니다 `Create AstraControlCenter` 양식 필드:

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 버전을 입력합니다(예: 23.04.2-7).

f. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.

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

h. 설치 scaleSize를 선택합니다.

기본적으로 Astra는 HA(High Availability)를 사용합니다. `scaleSize` 의 `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.

i. 수신 유형을 선택합니다.

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

  다른 수신 컨트롤러를 사용 중이거나 자체 수신 컨트롤러를 사용하려는 경우 이 옵션을 사용하십시오. Astra Control Center를 배포한 후 을 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. 이미지 레지스트리 * 에서 로컬 컨테이너 이미지 레지스트리 경로를 입력합니다. 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. 를 선택합니다 `Create`.

레지스트리 암호를 만듭니다

인증이 필요한 레지스트리를 사용하는 경우 OpenShift 클러스터에서 암호를 만들고 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 관리 Kubernetes 클러스터 중 하나에 Astra Control Center를 구축할 수 있습니다.

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

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

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

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

Amazon Web Services에 Astra Control Center를 구축합니다

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

AWS에 필요한 것

AWS에 Astra Control Center를 구축하기 전에 다음 항목이 필요합니다.

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

AWS의 운영 환경 요구사항

Astra Control Center에는 AWS를 위한 다음과 같은 운영 환경이 필요합니다.

- Red Hat OpenShift Container Platform 4.8

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

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

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

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

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

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

AWS 구축 개요

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

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. IAM 권한이 충분한지 확인하십시오.
2. AWS에 RedHat OpenShift 클러스터를 설치합니다.
3. AWS 구성.
4. 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 생성하고, ECR(Elastic Container Register)을 생성하여 Astra Control Center 이미지를 호스팅하고, 이 레지스트리로 이미지를 푸시합니다.

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

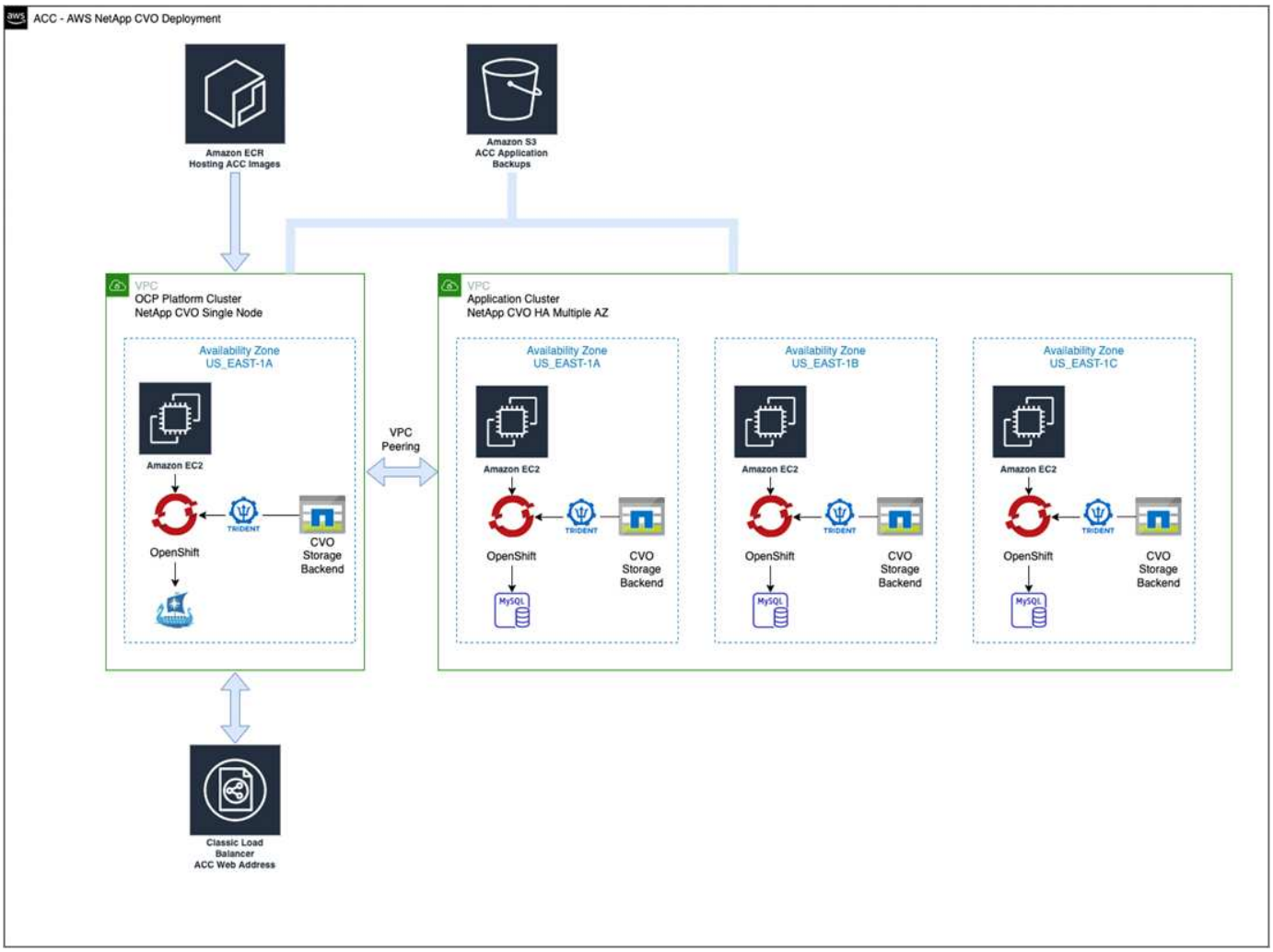
1. AWS 가상 네트워크를 생성합니다.
2. EC2 컴퓨팅 인스턴스를 검토합니다. 이는 AWS의 베어 메탈 서버 또는 VM이 될 수 있습니다.
3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 AWS의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 AWS S3 버킷을 하나 이상 생성합니다.
5. AWS ECR(Elastic Container Registry)을 생성하여 모든 ACC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

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

6. ACC 이미지를 정의된 레지스트리로 푸시합니다.

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

다음은 AWS 구축의 예입니다.



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 Trident 버전을 적어 둡니다.

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 Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

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

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

AWS용 **Astra Control Center**를 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) 4.10
- OCP를 사용하는 경우 Red Hat OpenShift Container Platform(OCP) 권한(네임스페이스 수준에서 POD 생성)
- 버킷 및 커넥터를 생성할 수 있는 권한이 있는 GCP 서비스 계정

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 요구 사항 외에 다음과 같은 리소스가 필요합니다.

구성 요소	요구 사항
백엔드 **NetApp Cloud Volumes ONTAP** 스토리지 용량입니다	최소 300GB가 사용 가능합니다

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

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

GCP 구축 개요

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

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. GCP에 RedHat OpenShift 클러스터를 설치합니다.
2. GCP 프로젝트 및 가상 프라이빗 클라우드를 생성합니다.

3. IAM 권한이 충분한지 확인하십시오.
4. GCP를 구성합니다.
5. NetApp 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

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

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

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

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

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

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

예:

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

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

8. DNS 존 설정

NetApp 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 모서리에서 Trident 버전을 확인합니다.
   c. "NetApp"을 프로비저닝자로 나타내는 Cloud Volumes ONTAP 클러스터 스토리지 클래스를 확인하십시오.

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

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

Astra Control Center 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 배포하기 전에 다음 항목이 필요합니다.

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

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 요구 사항 외에 다음과 같은 리소스가 필요합니다.

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

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

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

Azure 구축 개요

다음은 Azure용 Astra Control Center를 설치하는 프로세스의 개요입니다.

이러한 각 단계는 아래에 자세히 설명되어 있습니다.

1. Azure에 RedHat OpenShift 클러스터를 설치합니다.
2. Azure 리소스 그룹을 생성합니다.
3. IAM 권한이 충분한지 확인하십시오.
4. Azure를 구성합니다.
5. 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 만들고, Astra Control Center 이미지를 호스팅하기 위해 ACR(Azure Container Register)을 만들고, 이 레지스트리로 이미지를 푸시하도록 Azure를 구성합니다.

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

1. Azure 가상 네트워크를 생성합니다.
2. 컴퓨팅 인스턴스를 검토합니다. Azure의 베어 메탈 서버 또는 VM이 될 수 있습니다.

3. 인스턴스 유형이 마스터 및 작업자 노드에 대한 Astra 최소 리소스 요구 사항과 일치하지 않으면 Azure의 인스턴스 유형을 Astra 요구 사항에 맞게 변경합니다. 을 참조하십시오 "Astra Control Center 요구 사항".
4. 백업을 저장할 Azure Blob 컨테이너를 하나 이상 생성합니다.
5. 저장소 계정을 생성합니다. Astra Control Center에서 버킷으로 사용할 컨테이너를 생성하려면 저장소 계정이 필요합니다.
6. 버킷 액세스에 필요한 암호를 생성합니다.
7. Azure Container Registry(ACR)를 생성하여 모든 Astra Control Center 이미지를 호스트합니다.
8. Docker에 대한 ACR 액세스를 설정하여 모든 Astra Control Center 이미지를 푸시/풀합니다.
9. 다음 스크립트를 입력하여 ACC 이미지를 이 레지스트리에 푸시합니다.

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

◦ 예 *:

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

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

10. DNS 존 설정

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

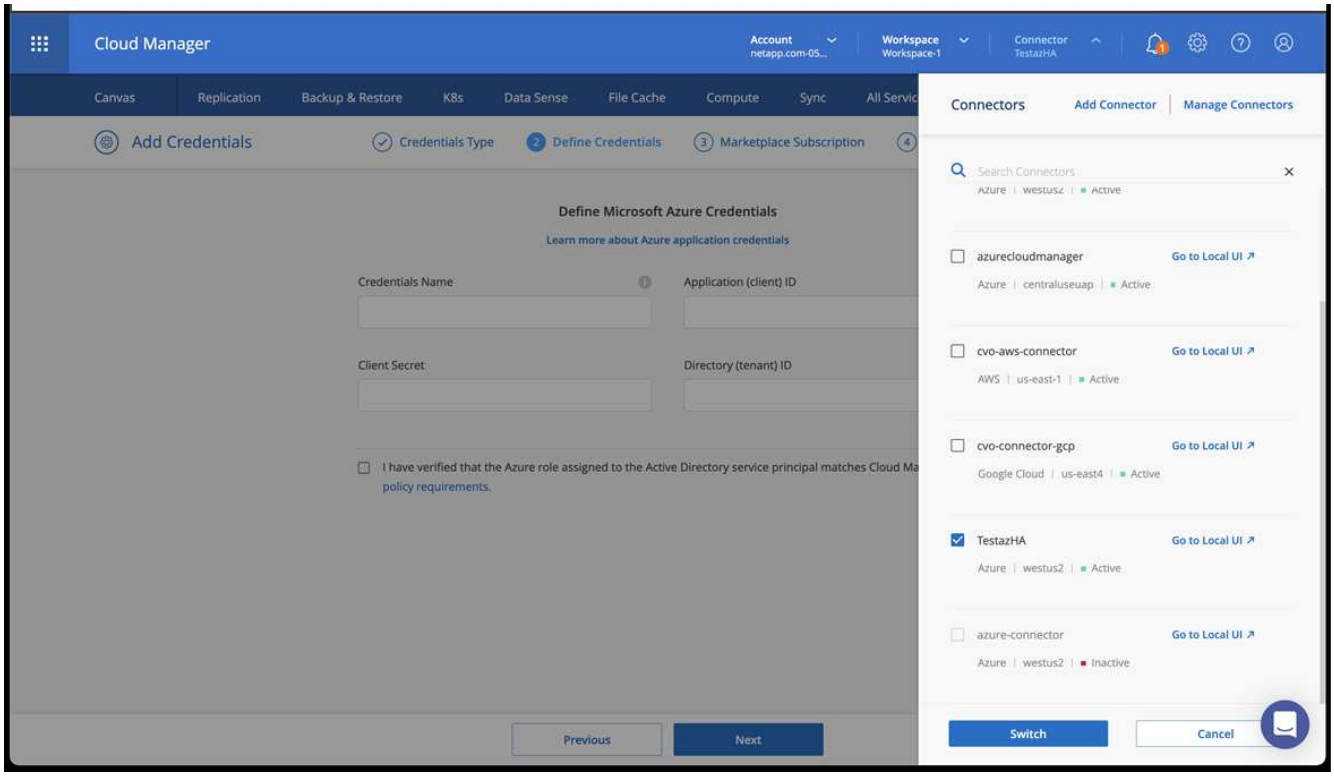
1. BlueXP에 자격 증명을 추가합니다.
2. Azure용 커넥터를 추가합니다. 을 참조하십시오 "BlueXP 정책".

a. 공급자로 * Azure * 를 선택합니다.

b. 애플리케이션 ID, 클라이언트 암호 및 디렉토리(테넌트) ID를 비롯한 Azure 자격 증명을 입력합니다.

을 참조하십시오 "BlueXPr에서 커넥터 만들기".

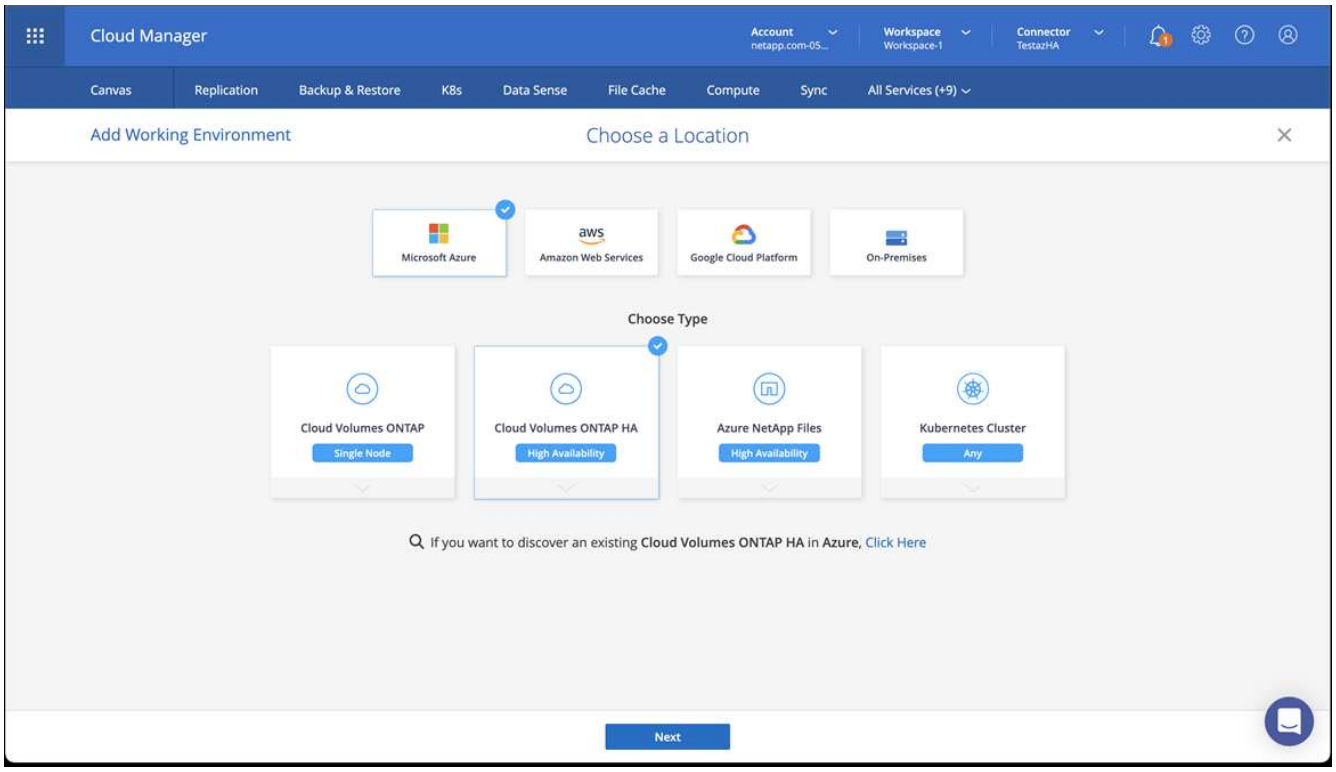
3. 커넥터가 실행 중인지 확인하고 해당 커넥터로 전환합니다.



4. 클라우드 환경을 위한 작업 환경을 구축합니다.

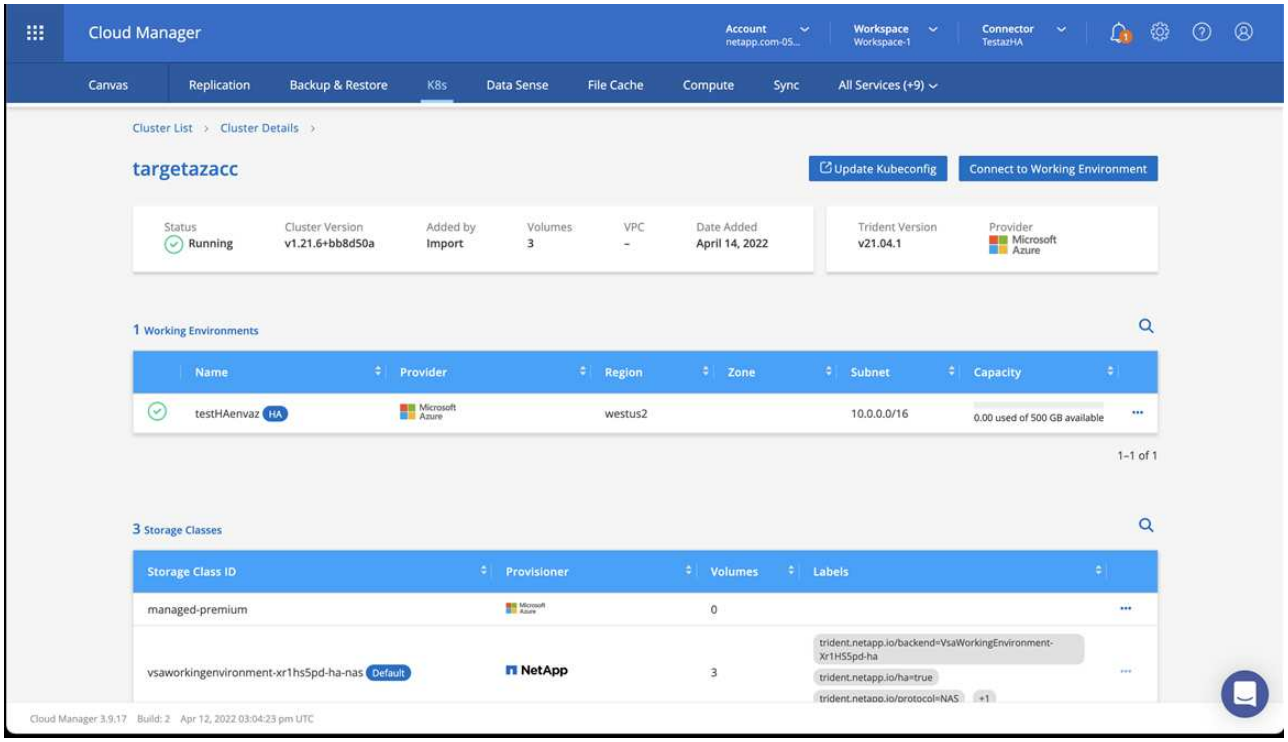
a. 위치: "Microsoft Azure".

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 Trident 버전을 적어 둡니다.

   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 Trident는 가져오기 및 검색 프로세스의 일부로 자동으로 설치됩니다.

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

Azure용 **Astra Control Center**를 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]
```

네임스페이스 간 네트워크 통신을 활성화합니다

일부 환경에서는 NetworkPolicy 구문을 사용하여 네임스페이스 간 트래픽을 제한합니다. Astra Control Center 운영자와 Astra Control Center는 서로 다른 네임스페이스에 있습니다. 서로 다른 네임스페이스에 있는 서비스는 서로 통신할 수 있어야 합니다. 이 통신을 활성화하려면 다음 단계를 수행하십시오.

단계

1. Astra Control Center 네임스페이스에 있는 NetworkPolicy 리소스를 삭제합니다.

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

2. 앞의 명령으로 반환된 각 NetworkPolicy 개체에 대해 다음 명령을 사용하여 개체를 삭제합니다. [OBJECT_NAME]을(를) 반환된 객체의 이름으로 바꿉니다.

```
kubectl delete networkpolicy [OBJECT_NAME] -n [netapp-acc or custom
namespace]
```

3. 다음 리소스 파일을 적용하여 를 구성합니다 `acc-avp-network-policy` Astra 플러그인 서비스가 Astra Control Center 서비스에 요청을 할 수 있도록 허용하는 개체입니다. 괄호 <>의 정보를 사용자 환경의 정보로 바꿉니다.

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

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

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

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

> 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
- 실행할 클러스터의 명령 셸에 대한 관리 액세스 `kubectl` 명령
- CA의 개인 키 및 인증서 파일

자체 서명된 인증서를 제거합니다

기존의 자체 서명된 TLS 인증서를 제거합니다.

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

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

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

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

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 를 변경합니다 `spec.selfSigned` 값을 로 설정합니다 `spec.ca.secretName` 앞에서 만든 TLS 암호를 확인하려면 다음을 수행하십시오.

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

#spec:
#  selfSigned: {}

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

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

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

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

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

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

5. 다음 명령을 사용하여 인증서를 생성합니다.

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

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

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

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

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

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

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

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

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