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"

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

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

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

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- 모든 클러스터 운영자가 양호한 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

  ```
  oc get clusteroperators
  ```

- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

  OpenShift 예:

  ```
  oc get apiservices
  ```

- 데이터 센터에 Astra Control Center에 대한 FQDN 주소를 만들었습니다.

이 작업에 대해

Astra Control Center 설치 프로세스는 다음을 수행합니다.

- "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스에 Astra 구성 요소를 설치합니다.
- 기본 계정을 만듭니다.
- Astra Control Center의 이 인스턴스에 대해 기본 관리 사용자 이메일 주소와 기본 1회 암호 ACC-<UUID_of_installation>(ACC-<UUID_of_installation>)를 설정합니다. 이 사용자에게는 시스템에서 소유자 역할이 할당되며 UI에 처음 로그인할 때 필요합니다.
- 모든 Astra Control Center Pod가 실행 중인지 확인하는 데 도움이 됩니다.
- Astra UI를 설치합니다.

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

모든 Astra Control Center POD를 삭제하지 않도록 설치 과정 내내 다음 명령을 실행하지 마십시오: "kubctl delete -f Astra_control_center_operator_deploy.YAML

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

- Astra Control Center 번들을 다운로드합니다
- 번들의 포장을 풀고 디렉토리를 변경합니다
- 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
- Astra Control Center UI에 로그인합니다

를 수행하여 배포를 완료합니다 "설정 작업".

Astra Control Center 번들을 다운로드합니다

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

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

번들의 포장을 풀고 디렉토리를 변경합니다

1. 이미지 추출:

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

2. Astra 디렉토리로 변경합니다.

```
cd astra-control-center-[version]
```

이미지를 로컬 레지스트리에 추가합니다

1. Astra Control Center 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.

   ⓘ 아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

   a. 레지스트리에 로그인합니다.

      Docker:

      ```
      docker login [your_registry_path]
      ```

      포드만:

      ```
      podman login [your_registry_path]
      ```

   b. 적절한 스크립트를 사용하여 이미지를 로드하고, 이미지에 태그를 지정하고, 를 눌러 이미지를 로컬 레지스트리에 푸시합니다.

      Docker:

      ```
      export REGISTRY=[Docker_registry_path]
      for astraImageFile in $(ls images/*.tar) ; do
        # Load to local cache. And store the name of the loaded image
      trimming the 'Loaded images: '
        astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
      image: //')
        astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
        # Tag with local image repo.
        docker tag ${astraImage} ${REGISTRY}/${astraImage}
        # Push to the local repo.
        docker push ${REGISTRY}/${astraImage}
      done
      ```

      포드만:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

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

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

   응답:

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

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

   ```
   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
   ```

   c. "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스를 생성합니다.

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

샘플 반응:

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

d. "NetApp-acc"(또는 사용자 지정 이름) 네임스페이스에 대한 암호를 생성합니다. 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. Astra Control Center 운영자 배포 YAML('Astra_control_center_operator_deploy.YAML')을 편집하여 현지 등록부와 비밀을 참조하십시오.

```
vim astra_control_center_operator_deploy.yaml
```

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

```
imagePullSecrets:
- name: <name_of_secret_with_creds_to_local_registry>
```

b. kuby-RBAC-proxy 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

c. "acc-operator-controller-manager" 이미지의 [your_registry_path]를 이미지를 에서 푸시한 레지스트리 경로로 변경합니다 이전 단계.

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

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

2. 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
```

Astra Control Center를 구성합니다

1. Astra Control Center 사용자 정의 리소스(CR) 파일('Astra_control_center_min YAML')을 편집하여 계정, AutoSupport, 레지스트리 및 기타 필요한 구성을 만듭니다.

> ⓘ 사용자 환경에 추가 사용자 정의가 필요한 경우 대체 CR로 Astra_control_center.yaML을 사용할 수 있습니다. Astra_control_center_min YAML은 기본 CR이며 대부분의 설치에 적합합니다.

```
vim astra_control_center_min.yaml
```

> ⓘ CR에서 구성한 속성은 초기 Astra Control Center 배포 후에는 변경할 수 없습니다.

> ⓘ 인증이 필요 없는 레지스트리를 사용하는 경우 imageRegistry 내에서 '비밀' 줄을 삭제해야 합니다. 그렇지 않으면 설치가 실패합니다.

a. '[your_registry_path]'를 이전 단계에서 이미지를 푸시한 레지스트리 경로로 변경합니다.

b. accountName 문자열을 계정과 연결할 이름으로 변경합니다.

c. Astra에 액세스하기 위해 브라우저에서 사용할 FQDN으로 "astraAddress" 문자열을 변경합니다. 주소에 http:// 또는 https:// 를 사용하지 마십시오. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.

d. e-메일 문자열을 기본 초기 관리자 주소로 변경합니다. 에서 사용할 이 이메일 주소를 복사합니다 나중에.

e. 인터넷 연결이 없는 사이트의 경우 AutoSupport에 등록된 사이트를 거짓으로 변경하거나 연결된 사이트의 경우 "참"으로 변경합니다.

f. (선택 사항) 계정과 연결된 사용자의 이름 "FirstName"과 성 "LastName"을 추가합니다. UI 내에서 이 단계를 지금 또는 나중에 수행할 수 있습니다.

g. (선택 사항) 설치에 필요한 경우 'torageClass' 값을 다른 Astra Trident StorageClass 리소스로 변경하십시오.

h. (Astra Data Store 미리 보기를 사용하여 설치하는 경우) 에 대해 알려진 문제를 참조하십시오 "추가 필수 변경 사항" YAML에 대한 정보를 제공합니다.

```
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"
```

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]
```

샘플 반응:

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

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

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

샘플 반응:

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

시스템 상태를 확인합니다

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

1. 모든 시스템 구성 요소가 성공적으로 설치되었는지 확인합니다.

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

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

샘플 반응:

```
NAME                                        READY   STATUS    RESTARTS
AGE
acc-helm-repo-5f75c5f564-bzqmt              1/1     Running   0
11m
activity-6b8f7cccb9-mlrn4                   1/1     Running   0
9m2s
api-token-authentication-6hznt              1/1     Running   0
8m50s
api-token-authentication-qpfgb              1/1     Running   0
8m50s
api-token-authentication-sqnb7              1/1     Running   0
8m50s
asup-5578bbdd57-dxkbp                       1/1     Running   0
9m3s
authentication-56bff4f95d-mspmq             1/1     Running   0
7m31s
bucketservice-6f7968b95d-9rrrl              1/1     Running   0
8m36s
cert-manager-5f6cf4bc4b-82khn               1/1     Running   0
6m19s
cert-manager-cainjector-76cf976458-sdrbc    1/1     Running   0
6m19s
cert-manager-webhook-5b7896bfd8-2n45j       1/1     Running   0
6m19s
cloud-extension-749d9f684c-8bdhq            1/1     Running   0
9m6s
cloud-insights-service-7d58687d9-h5tzw      1/1     Running   2
8m56s
composite-compute-968c79cb5-nv7l4           1/1     Running   0
```

```
9m11s
composite-volume-7687569985-jg9gg          1/1     Running   0
8m33s
credentials-5c9b75f4d6-nx9cz               1/1     Running   0
8m42s
entitlement-6c96fd8b78-zt7f8               1/1     Running   0
8m28s
features-5f7bfc9f68-gsjnl                  1/1     Running   0
8m57s
fluent-bit-ds-h88p7                        1/1     Running   0
7m22s
fluent-bit-ds-krhnj                        1/1     Running   0
7m23s
fluent-bit-ds-l5bjj                        1/1     Running   0
7m22s
fluent-bit-ds-lrclb                        1/1     Running   0
7m23s
fluent-bit-ds-s5t4n                        1/1     Running   0
7m23s
fluent-bit-ds-zpr6v                        1/1     Running   0
7m22s
graphql-server-5f5976f4bd-vbb4z            1/1     Running   0
7m13s
identity-56f78b8f9f-8h9p9                  1/1     Running   0
8m29s
influxdb2-0                                1/1     Running   0
11m
krakend-6f8d995b4d-5khkl                   1/1     Running   0
7m7s
license-5b5db87c97-jmxzc                   1/1     Running   0
9m
login-ui-57b57c74b8-6xtv7                  1/1     Running   0
7m10s
loki-0                                     1/1     Running   0
11m
monitoring-operator-9dbc9c76d-8znck        2/2     Running   0
7m33s
nats-0                                     1/1     Running   0
11m
nats-1                                     1/1     Running   0
10m
nats-2                                     1/1     Running   0
10m
nautilus-6b9d88bc86-h8kfb                  1/1     Running   0
8m6s
nautilus-6b9d88bc86-vn68r                  1/1     Running   0
```

```
8m35s
openapi-b87d77dd8-5dz9h                     1/1     Running   0
9m7s
polaris-consul-consul-5ljfb                 1/1     Running   0
11m
polaris-consul-consul-s5d5z                 1/1     Running   0
11m
polaris-consul-consul-server-0              1/1     Running   0
11m
polaris-consul-consul-server-1              1/1     Running   0
11m
polaris-consul-consul-server-2              1/1     Running   0
11m
polaris-consul-consul-twmpq                 1/1     Running   0
11m
polaris-mongodb-0                           2/2     Running   0
11m
polaris-mongodb-1                           2/2     Running   0
10m
polaris-mongodb-2                           2/2     Running   0
10m
polaris-ui-84dc87847f-zrg8w                 1/1     Running   0
7m12s
polaris-vault-0                             1/1     Running   0
11m
polaris-vault-1                             1/1     Running   0
11m
polaris-vault-2                             1/1     Running   0
11m
public-metrics-657698b66f-67pgt             1/1     Running   0
8m47s
storage-backend-metrics-6848b9fd87-w7x8r    1/1     Running   0
8m39s
storage-provider-5ff5868cd5-r9hj7           1/1     Running   0
8m45s
telegraf-ds-dw4hg                           1/1     Running   0
7m23s
telegraf-ds-k92gn                           1/1     Running   0
7m23s
telegraf-ds-mmxjl                           1/1     Running   0
7m23s
telegraf-ds-nhs8s                           1/1     Running   0
7m23s
telegraf-ds-rj7lw                           1/1     Running   0
7m23s
telegraf-ds-tqrkb                           1/1     Running   0
```

```
7m23s
telegraf-rs-9mwgj                           1/1     Running   0
7m23s
telemetry-service-56c49d689b-ffrzx          1/1     Running   0
8m42s
tenancy-767c77fb9d-g9ctv                    1/1     Running   0
8m52s
traefik-5857d87f85-7pmx8                    1/1     Running   0
6m49s
traefik-5857d87f85-cpxgv                    1/1     Running   0
5m34s
traefik-5857d87f85-lvmlb                    1/1     Running   0
4m33s
traefik-5857d87f85-t2xlk                    1/1     Running   0
4m33s
traefik-5857d87f85-v9wpf                    1/1     Running   0
7m3s
trident-svc-595f84dd78-zb8l6                1/1     Running   0
8m54s
vault-controller-86c94fbf4f-krttq           1/1     Running   0
9m24s
```

2. (선택 사항) 설치가 완료되었는지 확인하려면 다음 명령을 사용하여 "acc-operator" 로그를 볼 수 있습니다.

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

3. 모든 Pod가 실행 중인 경우, Astra Control Center Operator가 설치한 AstraControlCenter 인스턴스를 검색하여 설치 성공 여부를 확인한다.

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

4. '구축' 값에 대한 응답으로 'tatus.deploymentState` 필드를 확인합니다. 배포에 실패한 경우 대신 오류 메시지가 나타납니다.

> 다음 단계에서 uuid를 사용합니다.

```
name: astra
   namespace: netapp-acc
   resourceVersion: "104424560"
   selfLink: /apis/astra.netapp.io/v1/namespaces/netapp-
acc/astracontrolcenters/astra
   uid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
```

```
spec:
  accountName: Example
  astraAddress: astra.example.com
  astraVersion: 21.12.60
  autoSupport:
    enrolled: true
    url: https://support.netapp.com/asupprod/post/1.0/postAsup
  crds: {}
  email: admin@example.com
  firstName: SRE
  imageRegistry:
    name: registry_name/astra
    secret: astra-registry-cred
  lastName: Admin
status:
  accConditionHistory:
    items:
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:23:59Z"
        message: Deploying is currently in progress.
        reason: InProgress
        status: "False"
        type: Ready
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:23:59Z"
        message: Deploying is currently in progress.
        reason: InProgress
        status: "True"
```

```
        type: Deploying
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:23:59Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Post Install was successful
        observedGeneration: 2
        reason: Complete
        status: "True"
        type: PostInstallComplete
      generation: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Deploying succeeded.
        reason: Complete
```

```
        status: "False"
        type: Deploying
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Deployed
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
    - astraVersion: 21.12.60
```

```
      condition:
        lastTransitionTime: "2021-11-23T02:29:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Ready
      generation: 2
      observedGeneration: 2
      observedSpec:
        accountName: Example
        astraAddress: astra.example.com
        astraVersion: 21.12.60
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
          secret: astra-registry-cred
        lastName: Admin
      observedVersion: 21.12.60
      timestamp: "2021-11-23T02:29:41Z"
  certManager: deploy
  cluster:
    type: OCP
    vendorVersion: 4.7.5
    version: v1.20.0+bafe72f
  conditions:
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Astra is deployed
    reason: Complete
    status: "True"
    type: Ready
  - lastTransitionTime: "2021-12-08T16:19:55Z"
    message: Deploying succeeded.
    reason: Complete
    status: "False"
    type: Deploying
  - lastTransitionTime: "2021-12-08T16:19:53Z"
    message: Post Install was successful
    observedGeneration: 2
    reason: Complete
    status: "True"
    type: PostInstallComplete
```

```
    - lastTransitionTime: "2021-12-08T16:19:55Z"
      message: Astra is deployed
      reason: Complete
      status: "True"
      type: Deployed
    deploymentState: Deployed
    observedGeneration: 2
    observedSpec:
      accountName: Example
      astraAddress: astra.example.com
      astraVersion: 21.12.60
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
        secret: astra-registry-cred
      lastName: Admin
    observedVersion: 21.12.60
    postInstall: Complete
    uuid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
kind: List
metadata:
 resourceVersion: ""
 selfLink: ""
```

5. Astra Control Center에 로그인할 때 사용할 1회 암호를 얻으려면 이전 단계의 응답에서 'Status.uuid' 값을 복사합니다. 암호는 ACC-, UUID 값(ACC-[UUID]), 이 예에서는 ACC-c49008a5-4ef1-4c5d-a53e-830daf994116) 순으로 입력된다.

Astra Control Center UI에 로그인합니다

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

단계

1. 브라우저에서, Astra_control_center_min YAML'cr when의 astraAddress에 사용한 FQDN을 입력한다 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 Astra_control_center_min YAML CR when에 e-mail에 사용한 값을 입력합니다 Astra Control Center를 설치했습니다1회 암호('ACC-[UUID]')를 입력합니다.

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

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

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

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

설치 문제를 해결합니다

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

단계

1. Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

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

다음 단계

를 수행하여 배포를 완료합니다 "설정 작업".

OpenShift OperatorHub를 사용하여 Astra Control Center를 설치합니다

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

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

필요한 것

- "설치를 시작하기 전에 Astra Control Center 구축을 위한 환경을 준비합니다".
- OpenShift 클러스터에서 모든 클러스터 운영자가 정상 상태인지 확인합니다('사용 가능'은 '참'임).

```
oc get clusteroperators
```

- OpenShift 클러스터에서 모든 API 서비스가 정상 상태인지 확인합니다('사용 가능'은 '참'임).

```
oc get apiservices
```

- 데이터 센터에 Astra Control Center에 대한 FQDN 주소를 만들었습니다.
- 설명된 설치 단계를 수행하는 데 필요한 권한과 Red Hat OpenShift Container Platform에 대한 액세스 권한이 있습니다.

단계

- Astra Control Center 번들을 다운로드합니다
- 번들의 포장을 풀고 디렉토리를 변경합니다
- 이미지를 로컬 레지스트리에 추가합니다
- 운영자 설치 페이지를 찾으십시오
- 운전자를 설치합니다
- Astra Control Center를 설치합니다

Astra Control Center 번들을 다운로드합니다

1. 에서 Astra Control Center 번들('Astra-control-center-[version].tar.gz')을 다운로드합니다 "NetApp Support 사이트".
2. Astra Control Center 인증서 및 키의 지퍼를 에서 다운로드하십시오 "NetApp Support 사이트".
3. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

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

번들의 포장을 풀고 디렉토리를 변경합니다

1. 이미지 추출:

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

2. Astra 디렉토리로 변경합니다.

```
cd astra-control-center-[version]
```

이미지를 로컬 레지스트리에 추가합니다

1. Astra Control Center 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.

아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

a. 레지스트리에 로그인합니다.

Docker:

```
docker login [your_registry_path]
```

포드만:

```
podman login [your_registry_path]
```

b. 적절한 스크립트를 사용하여 이미지를 로드하고, 이미지에 태그를 지정하고, 를 눌러 이미지를 로컬 레지스트리에 푸시합니다.

Docker:

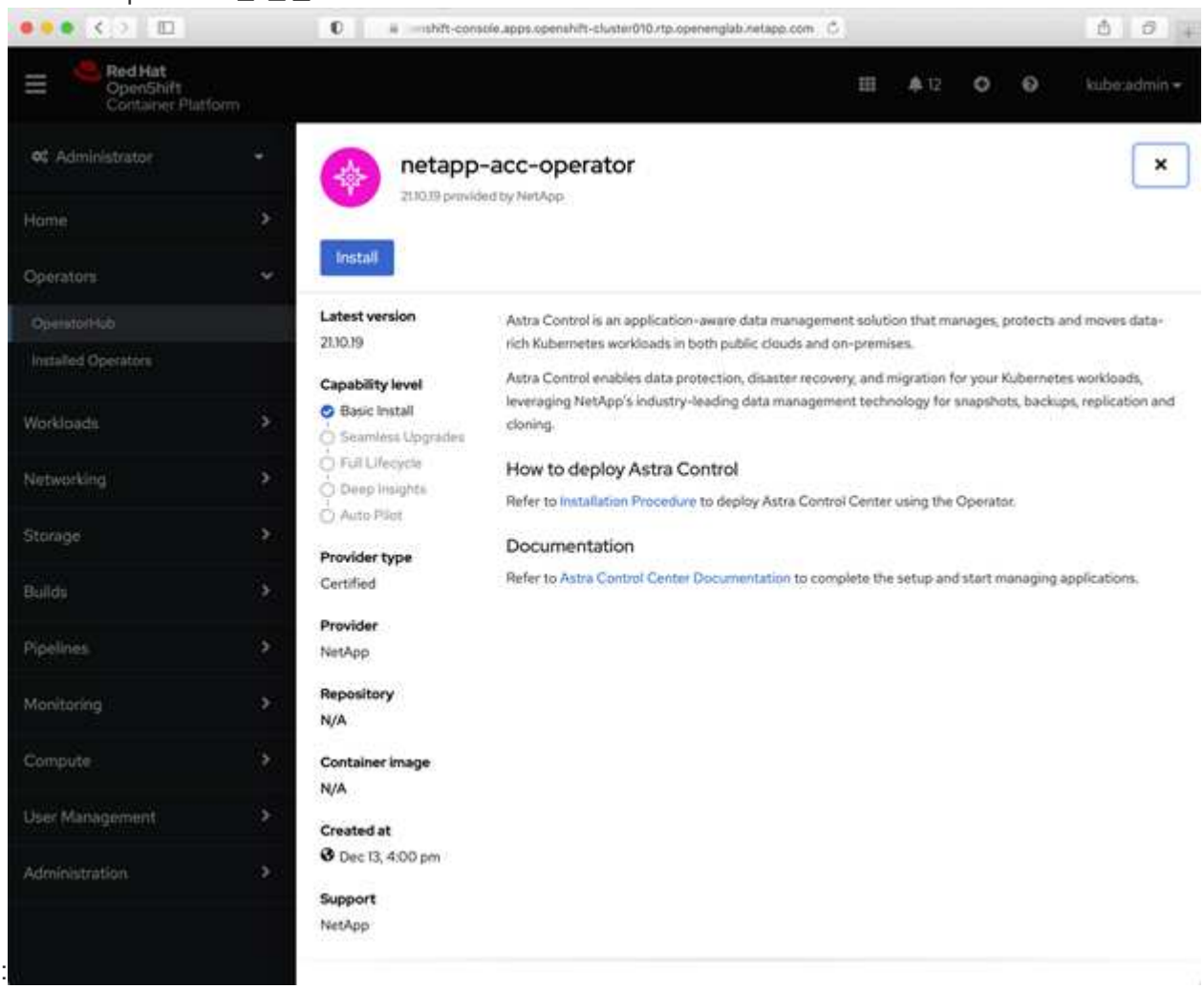
```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image
trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

포드만:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded image trimming
the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

운영자 설치 페이지를 찾으십시오

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



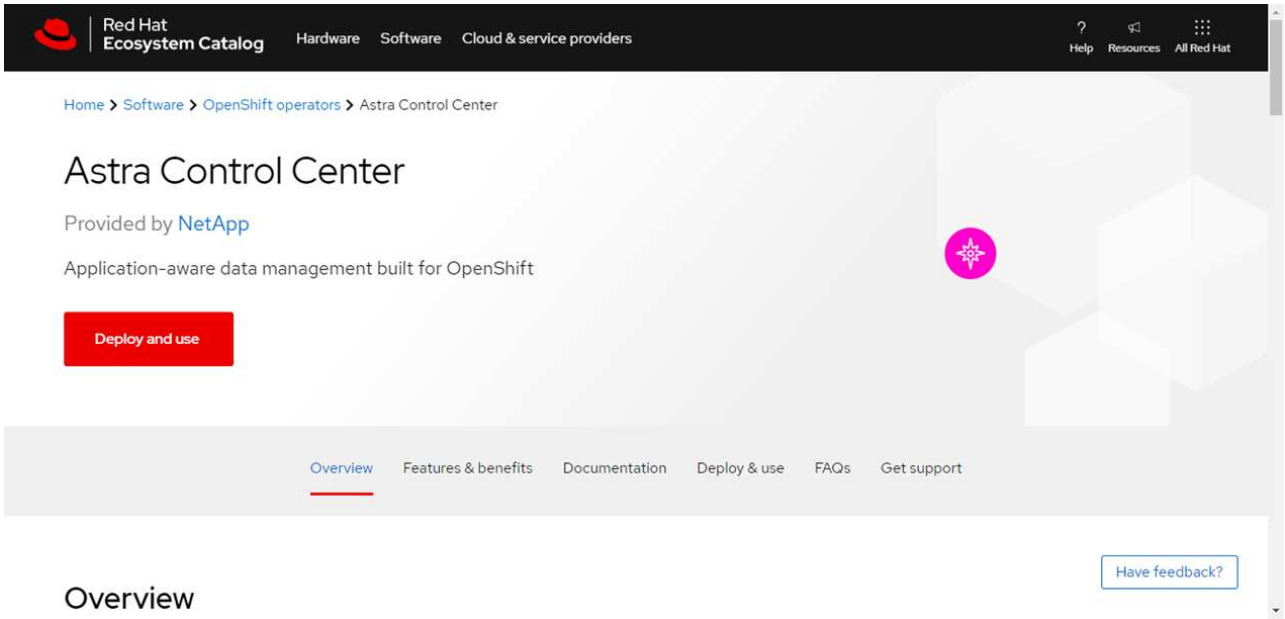
     i. OpenShift Container Platform UI에 로그인합니다.

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

iii. NetApp Astra Control Center 운영자를 선택합니다.

iv. 설치 * 를 선택합니다.

- Red Hat 에코시스템 카탈로그
:



i. NetApp Astra Control Center를 선택합니다 "운영자".

ii. 배포 및 사용 * 을 선택합니다.

운전자를 설치합니다

1. Install Operator * 페이지를 완료하고 운영자를 설치합니다.

> 운영자는 모든 클러스터 네임스페이스에서 사용할 수 있습니다.

a. 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 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 운용자의 상세보기 내의 콘솔에서 제공된 API 섹션에서 'Create instance'를 선택한다.

2. 'Create AstraControlCenter' 양식 필드를 작성합니다.
    a. Astra Control Center 이름을 유지하거나 조정합니다.
    b. (선택 사항) 자동 지원을 활성화 또는 비활성화합니다. 자동 지원 기능을 유지하는 것이 좋습니다.
    c. Astra Control Center 주소를 입력합니다. 주소에 http:// 또는 https:// 를 입력하지 마십시오.
    d. Astra Control Center 버전을 입력합니다(예: 21.12.60).
    e. 계정 이름, 이메일 주소 및 관리자 성을 입력합니다.
    f. 기본 볼륨 재확보 정책을 유지합니다.
    g. 이미지 레지스트리 * 에서 로컬 컨테이너 이미지 레지스트리 경로를 입력합니다. 주소에 http:// 또는 https:// 를 입력하지 마십시오.
    h. 인증이 필요한 레지스트리를 사용하는 경우 암호를 입력합니다.
    i. 관리자의 이름을 입력합니다.
    j. 리소스 확장을 구성합니다.
    k. 기본 스토리지 클래스를 유지합니다.
    l. CRD 처리 기본 설정을 정의합니다.
3. Create를 선택합니다.

다음 단계

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