Astra Control Center를 설치합니다

Astra Control Center

NetApp
November 21, 2023

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

목차

표준 프로세스를 사용하여 Astra Control Center를 설치합니다 . . . 1
Astra Control Center 번들을 다운로드하고 포장을 풉니다 . . . 2
NetApp Astra kubtl 플러그인을 설치합니다 . . . 2
이미지를 로컬 레지스트리에 추가합니다 . . . 3
인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다 . . . 5
Astra Control Center 운영자를 설치합니다 . . . 7
Astra Control Center를 구성합니다 . . . 9
Astra 제어 센터 및 운전자 설치를 완료합니다 . . . 11
시스템 상태를 확인합니다 . . . 12
부하 분산을 위한 수신 설정 . . . 16
Astra Control Center UI에 로그인합니다 . . . 21
설치 문제를 해결합니다 . . . 22
다음 단계 . . . 22
POD 보안 정책 제한 사항 이해 . . . 22

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

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

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

필요한 것

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

```
kubectl get clusteroperators
```

- 모든 API 서비스가 정상 상태이며 사용 가능한지 확인합니다.

```
kubectl get apiservices
```

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

이 작업에 대해

Astra Control Center 설치 프로세스는 다음을 수행합니다.

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

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

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

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

단계

Astra Control Center를 설치하려면 다음 단계를 수행하십시오.

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

Astra Control Center 번들을 다운로드하고 포장을 풉니다

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

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

4. 이미지 추출:

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

NetApp Astra kubtl 플러그인을 설치합니다

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

필요한 것

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

단계

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

```
ls kubectl-astra/
```

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

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

이미지를 로컬 레지스트리에 추가합니다

1. 용기 엔진에 적합한 단계 시퀀스를 완료합니다.

Docker 를 참조하십시오

1. Astra 디렉토리로 이동합니다.

```
cd acc
```

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

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

팟맨

1. 레지스트리에 로그인합니다.

```
podman login [your_registry_path]
```

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

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

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

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

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

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

인증 요구 사항이 있는 레지스트리에 대한 네임스페이스 및 암호를 설정합니다

1. Astra Control Center 호스트 클러스터에 대한 KUBECONFIG를 내보냅니다.

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

2. 인증이 필요한 레지스트리를 사용하는 경우 다음을 수행해야 합니다.

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

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

응답:

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

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

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

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

샘플 반응:

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

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

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

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

샘플 반응:

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

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

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

응답

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

a. [[substep_kubecononfig_secret] (선택 사항) 설치 후 Astra Control Center에서 클러스터를 자동으로 관리하려는 경우 이 명령을 사용하여 배포할 Astra Control Center 네임스페이스 내에서 kubecononfig를 암호로 제공해야 합니다.

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

Astra Control Center 운영자를 설치합니다

1. 디렉토리를 변경합니다.

```
cd manifests
```

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

```
vim astra_control_center_operator_deploy.yaml
```

> YAML 주석이 붙은 샘플은 다음 단계를 따릅니다.

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

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

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

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

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

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

3. Astra Control Center 운영자를 설치합니다.

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

샘플 반응:

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

4. Pod가 실행 중인지 확인합니다.

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

Astra Control Center를 구성합니다

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

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

```
vim astra_control_center_min.yaml
```

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

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

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

c. 를 변경합니다 `astraAddress` 브라우저에서 Astra에 액세스하기 위해 사용할 FQDN에 대한 문자열입니다. 사용하지 마십시오 `http://` 또는 `https://` 를 입력합니다. 에서 사용하기 위해 이 FQDN을 복사합니다 나중에.

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

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

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

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

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

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

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

j. 다음 단계 중 하나를 수행합니다.

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

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

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

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

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

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

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

Astra 제어 센터 및 운전자 설치를 완료합니다

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

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

샘플 반응:

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

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

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

샘플 반응:

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

시스템 상태를 확인합니다

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

1. 모든 시스템 구성 요소가 성공적으로 설치되었는지 확인합니다.

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

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

샘플 응답

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

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

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

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

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

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

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

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

응답:

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

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

부하 분산을 위한 수신 설정

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

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

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

단계는 사용하는 수신 컨트롤러의 유형에 따라 다릅니다.

- 이스티오 침투
- Nginx 수신 컨트롤러
- OpenShift 수신 컨트롤러

필요한 것

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

Istio 침투에 대한 단계

1. Istio Ingress를 구성합니다.

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

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

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

   명령 예:

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

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

   명령 예:

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

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

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

   출력:

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

v1 새 스키마의 경우 다음 샘플을 따르십시오.

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

출력:

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

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

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

응답:

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

Nginx 수신 컨트롤러 단계

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

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

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

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

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

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

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

OpenShift Ingress 컨트롤러를 위한 단계

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

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

Astra Control Center UI에 로그인합니다

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

단계

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

> 로그인 후 사용자 지정 인증서를 만들 수 있습니다.

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

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

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

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

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

설치 문제를 해결합니다

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

단계

1. Astra Control Center 운영자 로그를 검사하려면 다음을 입력하십시오.

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

다음 단계

를 수행하여 배포를 완료합니다 "설정 작업".

=
:allow-uri-read:

POD 보안 정책 제한 사항 이해

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

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

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

Astra Control Center에서 설치한 PSP

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

설치 중에 PSP가 생성되었습니다

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

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

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

kubectl get role

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

kubectl get rolebinding

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

백업 작업 중에 PSP가 생성되었습니다

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

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

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

kubectl get role

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

kubectl get rolebinding

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

클러스터 관리 중에 생성된 **PSP**입니다

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

새 정책 및 객체에는 다음과 같은 특성이 있습니다.

```
kubectl get psp

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

kubectl get role

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

kubectl get rolebinding

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

네임스페이스 간 네트워크 통신을 활성화합니다

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

단계

1. Astra Control Center 네임스페이스에 있는 NetworkPolicy 리소스를 삭제합니다.

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

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

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

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

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

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

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

리소스 제한을 제거합니다

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

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

단계

1. NetApp-acc 네임스페이스에서 리소스 할당량 확인:

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

응답:

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

2. 이름으로 모든 리소스 할당량 삭제:

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

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

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

3. NetApp-acc 네임스페이스의 제한 범위를 가져옵니다.

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

응답:

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

4. 이름별로 제한 범위를 삭제합니다.

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

=
:allow-uri-read:

저작권 정보

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

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

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

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

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

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

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

상표 정보

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