Install Astra Control Center

Astra Control Center

NetApp
November 30, 2022

Table of Contents

Install Astra Control Center using the standard process

To install Astra Control Center, download the installation bundle from the NetApp Support Site and perform the following steps. You can use this procedure to install Astra Control Center in internet-connected or air-gapped environments.

Other installation procedures

- **Install with RedHat Openshift OperatorHub**: Use this alternative procedure to install Astra Control Center on Openshift using OperatorHub.
- **Install in the public cloud with Cloud Volumes ONTAP backend**: Use these procedures to install Astra Control Center in Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure with a Cloud Volumes ONTAP storage backend.

What you'll need

- Before you begin installation, prepare your environment for Astra Control Center deployment.
- If you have configured or want to configure pod security policies in your environment, familiarize yourself with pod security policies and how they affect Astra Control Center installation. See Understand pod security policy restrictions.
- Ensure all API services are in a healthy state and available:

```
kubectl get apiservices
```

- Ensure the Astra FQDN you plan to use is routable to this cluster. This means that you either have a DNS entry in your internal DNS server or you are using a core URL route that is already registered.
- If a cert-manager already exists in the cluster, you need to perform some prerequisite steps so that Astra Control Center does not attempt to install its own cert-manager. By default, Astra Control Center installs its own cert-manager during installation.

About this task

The Astra Control Center installation process helps you to do the following:

- Install the Astra components into the `netapp-acc` (or custom-named) namespace.
- Create a default Astra Control Owner admin account.
- Establish an administrative user email address and default initial setup password. This user is assigned the Owner role that is needed for first time login to the UI.
- Determine that all Astra Control Center pods are running.
- Install the Astra Control Center UI.

Do not delete the Astra Control Center operator (for example, `kubectl delete -f astra_control_center_operator_deploy.yaml`) at any time during Astra Control Center installation or operation to avoid deleting pods.

Steps

To install Astra Control Center, do the following steps:

- Download and unpack the Astra Control Center bundle
- Install the NetApp Astra kubectl plugin
- Add the images to your local registry
- Set up namespace and secret for registries with auth requirements
- Install the Astra Control Center operator
- Configure Astra Control Center
- Complete Astra Control Center and operator installation
- Verify system status
- Set up ingress for load balancing
- Log in to the Astra Control Center UI

Download and unpack the Astra Control Center bundle

1. Download the Astra Control Center bundle (`astra-control-center-[version].tar.gz`) from the NetApp Support Site.
2. Download the zip of Astra Control Center certificates and keys from the NetApp Support Site.
3. (Optional) Verify the signature of the bundle:

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

4. Extract the images:

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

Install the NetApp Astra kubectl plugin

The NetApp Astra kubectl command line plugin saves time when performing common tasks associated with deploying and upgrading Astra Control Center.

What you'll need

NetApp provides plugin binaries for different CPU architectures and operating systems. You need to know which CPU and operating system you have before you perform this task.

Steps

1. List the available NetApp Astra kubectl plugin binaries, and note the name of the file you need for your operating system and CPU architecture:

The kubectl plugin library is part of the tar bundle and is extracted into the folder `kubectl-astra`.

```
ls kubectl-astra/
```

2. Move the correct binary into the current path and rename it to `kubectl-astra`:

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

Add the images to your local registry

1. Complete the appropriate step sequence for your container engine:

Docker

a. Change to the `acc` directory extracted from the tar bundle:

   Example:

```
cd ../acc
```

a. Push the package images in the Astra Control Center image directory to your local registry. Make the following substitutions before running the command:

   - Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.yaml`).
   - Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, https://exampledownloads.jfrog.io/docker-astra-control/v1/.
   - Replace <MY_REGISTRY_USER> with the user name.
   - Replace <MY_REGISTRY_TOKEN> with an authorized token for the registry.

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

Podman

a. Log in to your registry:

```
podman login <MY_FULL_REGISTRY_PATH>
```

b. Run the following script, making the <YOUR_REGISTRY> substitution as noted in the comments:

```
# 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=22.11.0-82
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
```

Set up namespace and secret for registries with auth requirements

1. Export the KUBECONFIG for the Astra Control Center host cluster:

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

> Before you complete the installation, be sure your KUBECONFIG is pointing to the cluster where you want to install Astra Control Center. The KUBECONFIG can contain only one context.

2. If you use a registry that requires authentication, you need to do the following:
   a. Create the `netapp-acc-operator` namespace:

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

Response:

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

b. Create a secret for the `netapp-acc-operator` namespace. Add Docker information and run the following command:

> The placeholder `your_registry_path` should match the location of the images that you uploaded earlier (for example, `[Registry_URL]/netapp/astra/astracc/22.11.0-82`).

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

Sample response:

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

> If you delete the namespace after the secret is generated, recreate the namespace and then regenerate the secret for the namespace.

c. Create the `netapp-acc` (or custom-named) namespace.

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

Sample response:

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

d. Create a secret for the `netapp-acc` (or custom-named) namespace. Add Docker information and run the following command:

```
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]
```

Response

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

Install the Astra Control Center operator

1. Change the directory:

```
cd manifests
```

2. Edit the Astra Control Center operator deployment YAML (`astra_control_center_operator_deploy.yaml`) to refer to your local registry and secret.

```
vim astra_control_center_operator_deploy.yaml
```

An annotated sample YAML follows these steps.

a. If you use a registry that requires authentication, replace the default line of `imagePullSecrets: []` with the following:

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

b. Change `[your_registry_path]` for the `kube-rbac-proxy` image to the registry path where you pushed the images in a previous step.

c. Change `[your_registry_path]` for the `acc-operator-controller-manager` image to the registry path where you pushed the images in a previous step.

```
<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:[version x.y.z]
        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. Install the Astra Control Center operator:

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

Sample response:

```
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. Verify pods are running:

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

Configure Astra Control Center

1. Edit the Astra Control Center custom resource (CR) file (`astra_control_center.yaml`) to make account, AutoSupport, registry, and other necessary configurations:

For additional customizations, familiarize yourself with all CR options and their potential values to ensure you deploy Astra Control Center correctly for your environment.

```
vim astra_control_center.yaml
```

An annotated sample YAML follows these steps.

a. **accountName**: Change the `accountName` string to the name you want to associate with the account.

b. **astraAddress**: Change the `astraAddress` string to the FQDN (recommended) or IP address you want to use in your browser to access Astra Control Center. This is the same FQDN or IP address you provisioned from your load balancer when you completed Astra Control Center requirements.

   Do not use `http://` or `https://` in the address. Copy this FQDN for use in a later step.

c. **autoSupport**: Change `enrolled` for AutoSupport to `false` for sites without internet connectivity or retain `true` for connected sites.

d. **email**: Change the `email` string to the default initial administrator address. Copy this email address for use in a later step.

e. **(Optional) firstName and LastName**: Add a first name `firstName` and last name `lastName` of the user associated with the account. You can perform this step now or later within the UI.

f. **imageRegistry**: Change `[your_registry_path]` to the registry path where you pushed the images in the previous step.

   If you are using a registry that does not require authorization, you must delete the `secret` line within `imageRegistry` or the installation will fail.

g. **(Optional) storageClass**: Change the `storageClass` value from 'ontap-gold' to another Trident storageClass resource as required by your installation. Run the command `kubectl get sc` to determine your existing configured storage classes.

h. **ingressType**: Use one of the following ingress types:

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

     Use this option when you have another ingress controller in use or would prefer to use your own ingress controller. After Astra Control Center is deployed, you will need to configure the ingress controller to expose Astra Control Center with a URL.

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

     Use this option when you would prefer not to configure an ingress controller. This deploys the Astra Control Center `traefik` gateway as a Kubernetes LoadBalancer type service.

     Astra Control Center uses a service of the type "LoadBalancer" (`svc/traefik` in the Astra Control Center namespace), and requires that it be assigned an accessible external IP address. If load balancers are permitted in your environment and you don't already have one configured, you can use MetalLB or another external service load balancer to assign an external IP address to the

service. In the internal DNS server configuration, you should point the chosen DNS name for Astra Control Center to the load-balanced IP address.

> For details about the service type of "LoadBalancer" and ingress, see Requirements.

i. **crds**: If you use an external cert-manager, change `externalCertManager` to `true`. The default `false` causes Astra Control Center to install its own cert-manager during installation.

```
<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"
  astraResourcesScaler: "Default"
  additionalValues: {}
  crds:
    externalTraefik: false
    externalCertManager: false
```

Complete Astra Control Center and operator installation

1. If you didn't already do so in a previous step, create the `netapp-acc` (or custom) namespace:

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

Sample response:

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

2. Install Astra Control Center in the `netapp-acc` (or your custom) namespace:

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

Sample response:

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

Verify system status

You can verify system status using kubectl commands. If you prefer to use OpenShift, you can use comparable oc commands for verification steps.

Steps

1. Verify that all system components installed successfully.

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

Each pod should have a status of `Running`. It may take several minutes before the system pods are deployed.

Sample response

```
NAME                                          READY   STATUS
RESTARTS        AGE
acc-helm-repo-76d8d845c9-ggds2                1/1     Running   0
14m
activity-6cc67ff9f4-z48mr                     1/1     Running   2
(8m32s ago)   9m
api-token-authentication-7s67v                1/1     Running   0
8m56s
api-token-authentication-bplb4                1/1     Running   0
8m56s
api-token-authentication-p2c9z                1/1     Running   0
8m56s
asup-6cdfbc6795-md8vn                         1/1     Running   0
9m14s
authentication-9477567db-8hnc9                1/1     Running   0
7m4s
bucketservice-f4dbdfcd6-wqzkw                 1/1     Running   0
8m48s
cert-manager-bb756c7c4-wm2cv                  1/1     Running   0
14m
cert-manager-cainjector-c9bb86786-8wrf5       1/1     Running   0
14m
cert-manager-webhook-dd465db99-j2w4x          1/1     Running   0
14m
certificates-68dff9cdd6-kcvml                 1/1     Running   2
(8m43s ago)   9m2s
certificates-68dff9cdd6-rsnsb                 1/1     Running   0
9m2s
cloud-extension-69d48c956c-2s8dt              1/1     Running   3
(8m43s ago)   9m24s
cloud-insights-service-7c4f48b978-7gvlh       1/1     Running   3
(8m50s ago)   9m28s
composite-compute-7d9ff5f68-nxbhl             1/1     Running   0
8m51s
composite-volume-57b4756d64-nl66d             1/1     Running   0
9m13s
credentials-6dbc55f89f-qpzff                  1/1     Running   0
11m
entitlement-67bfb6d7-gl6kp                    1/1     Running   4
(8m33s ago)   9m38s
features-856cc4dccc-mxbdb                     1/1     Running   0
9m20s
fluent-bit-ds-4rtsp                           1/1     Running   0
6m54s
```

```
fluent-bit-ds-9rqll                          1/1     Running   0
6m54s
fluent-bit-ds-w5mp7                          1/1     Running   0
6m54s
graphql-server-7c7cc49776-jz2kn              1/1     Running   0
2m29s
identity-87c59c975-9jpnf                     1/1     Running   0
9m6s
influxdb2-0                                  1/1     Running   0
13m
keycloak-operator-84ff6d59d4-qcnmc           1/1     Running   0
7m1s
krakend-cbf6c7df9-mdtzv                      1/1     Running   0
2m30s
license-5b888b78bf-plj6j                     1/1     Running   0
9m32s
login-ui-846b4664dd-fz8hv                    1/1     Running   0
2m24s
loki-0                                       1/1     Running   0
13m
metrics-facade-779cc9774-n26rw               1/1     Running   0
9m18s
monitoring-operator-974db78f-pkspq           2/2     Running   0
6m58s
nats-0                                       1/1     Running   0
13m
nats-1                                       1/1     Running   0
13m
nats-2                                       1/1     Running   0
13m
nautilus-7bdc7ddc54-49tfn                    1/1     Running   0
7m50s
nautilus-7bdc7ddc54-cwc79                    1/1     Running   0
9m36s
openapi-5584ff9f46-gbrdj                     1/1     Running   0
9m17s
openapi-5584ff9f46-z9mzk                     1/1     Running   0
9m17s
packages-bfc58cc98-lpxq9                     1/1     Running   0
8m58s
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   3
(6m15s ago)   6m56s
polaris-keycloak-1                          1/1     Running   0
4m22s
polaris-keycloak-2                          1/1     Running   0
3m41s
polaris-keycloak-db-0                       1/1     Running   0
6m56s
polaris-keycloak-db-1                       1/1     Running   0
4m23s
polaris-keycloak-db-2                       1/1     Running   0
3m36s
polaris-mongodb-0                           2/2     Running   0
13m
polaris-mongodb-1                           2/2     Running   0
13m
polaris-mongodb-2                           2/2     Running   0
12m
polaris-ui-5ccff47897-8rzgh                 1/1     Running   0
2m33s
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-6cb7bfc49b-p54xm             1/1     Running   1
(8m29s ago)   9m31s
storage-backend-metrics-5c77994586-kjn48    1/1     Running   0
8m52s
storage-provider-769fdc858c-62w54           1/1     Running   0
8m54s
task-service-9ffc484c5-kx9f4                1/1     Running   3
(8m44s ago)   9m34s
telegraf-ds-bphb9                           1/1     Running   0
6m54s
telegraf-ds-rtsm2                           1/1     Running   0
6m54s
telegraf-ds-s9h5h                           1/1     Running   0
6m54s
telegraf-rs-lbpv7                           1/1     Running   0
6m54s
telemetry-service-57cfb998db-zjx78          1/1     Running   1
(8m40s ago)   9m26s
tenancy-5d5dfbcf9f-vmbxh                    1/1     Running   0
9m5s
```

```
traefik-7b87c4c474-jmgp2                  1/1     Running   0
2m24s
traefik-7b87c4c474-t9k8x                  1/1     Running   0
2m24s
trident-svc-c78f5b6bd-nwdsq               1/1     Running   0
9m22s
vault-controller-55bbc96668-c6425         1/1     Running   0
11m
vault-controller-55bbc96668-lq9n9         1/1     Running   0
11m
vault-controller-55bbc96668-rfkgg         1/1     Running   0
11m
```

2. (Optional) To ensure the installation is completed, you can watch the `acc-operator` logs using the following command.

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

> `accHost` cluster registration is one of the last operations, and if it fails it will not cause deployment to fail. In the event of a cluster registration failure indicated in the logs, you can attempt registration again through the Add cluster workflow in the UI or API.

3. When all the pods are running, verify that the installation was successful (`READY` is `True`) and get the initial setup password you will use when you log in to Astra Control Center:

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

Response:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   22.11.0-82  10.111.111.111
True
```

> Copy the UUID value. The password is `ACC-` followed by the UUID value (`ACC-[UUID]` or, in this example, `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`).

Set up ingress for load balancing

You can set up a Kubernetes ingress controller that manages external access to services. These procedures give setup examples for an ingress controller if you used the default of `ingressType: "Generic"` in the

Astra Control Center custom resource (`astra_control_center.yaml`). You do not need to use this procedure if you specified `ingressType: "AccTraefik"` in the Astra Control Center custom resource (`astra_control_center.yaml`).

After Astra Control Center is deployed, you will need to configure the ingress controller to expose Astra Control Center with a URL.

Setup steps differ depending on the type of ingress controller you use. Astra Control Center supports many ingress controller types. These setup procedures provide example steps for the following ingress controller types:

- Istio ingress
- Nginx ingress controller
- OpenShift ingress controller

What you'll need

- The required ingress controller should already be deployed.
- The ingress class corresponding to the ingress controller should already be created.

Steps for Istio ingress

1. Configure Istio ingress.

   This procedure assumes that Istio is deployed using the "default" configuration profile.

2. Gather or create the desired certificate and private key file for the Ingress Gateway.

   You can use a CA-signed or self-signed certificate. The common name must be the Astra address (FQDN).

   Sample command:

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

3. Create a secret `tls secret name` of type `kubernetes.io/tls` for a TLS private key and certificate in the `istio-system namespace` as described in TLS secrets.

   Sample command:

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

   The name of the secret should match the `spec.tls.secretName` provided in `istio-ingress.yaml` file.

4. Deploy an ingress resource in the `netapp-acc` (or custom-named) namespace using the v1 resource type for a schema (`istio-Ingress.yaml` is used in this example):

```
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 addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. Apply the changes:

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

6. Check the status of the ingress:

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

Response:

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

7. Finish Astra Control Center installation.

Steps for Nginx ingress controller

1. Create a secret of type `kubernetes.io/tls` for a TLS private key and certificate in `netapp-acc` (or custom-named) namespace as described in TLS secrets.
2. Deploy an ingress resource in `netapp-acc` (or custom-named) namespace using the v1 resource type for a schema (`nginx-Ingress.yaml` is used in this example):

```
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
```

3. Apply the changes:

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

NetApp recommends installing the nginx controller as a deployment rather than a daemonSet.

Steps for OpenShift ingress controller

1. Procure your certificate and get the key, certificate, and CA files ready for use by the OpenShift route.
2. Create the OpenShift route:

```
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
```

Log in to the Astra Control Center UI

After installing Astra Control Center, you will change the password for the default administrator and log in to the Astra Control Center UI dashboard.

Steps

1. In a browser, enter the FQDN (<code><a href="https://<FQDN&gt" class="bare">https://<FQDN&gt;</code>) you used in the <code>astraAddress</code> in the <code>astra_control_center.yaml</code> CR when you installed Astra Control Center.
2. Accept the self-signed certificates if prompted.

   You can create a custom certificate after login.

3. At the Astra Control Center login page, enter the value you used for `email` in `astra_control_center.yaml` CR when you installed Astra Control Center, followed by the initial setup password (`ACC-[UUID]`).

   If you enter an incorrect password three times, the admin account will be locked for 15 minutes.

4. Select **Login**.
5. Change the password when prompted.

   If this is your first login and you forget the password and no other administrative user accounts have yet been created, contact NetApp Support for password recovery assistance.

6. (Optional) Remove the existing self-signed TLS certificate and replace it with a custom TLS certificate signed by a Certificate Authority (CA).

Troubleshoot the installation

If any of the services are in `Error` status, you can inspect the logs. Look for API response codes in the 400 to 500 range. Those indicate the place where a failure happened.

Steps

1. To inspect the Astra Control Center operator logs, enter the following:

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

What's next

- (Optional) Depending on your environment, complete post-installation configuration steps.
- Complete the deployment by performing setup tasks.

Astra Control Center cluster custom resource options

You can use the following Astra Control Center cluster custom resource (CR) options to create custom configurations during deployment.

Setting	Type	Use	Value Example	Description
accountName	string	Required	Example	Astra Control Center account name. There can be only one.
astraAddress	string	Required	astra.example.com	Defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center.
astraKubeConfigSecret	string	Required	acc-kubeconfig-cred	If this value is present and a secret exists, the operator will attempt to add that KubeConfig to become the first managed cluster.
astraResourcesScaler	string	Required	Default	Scaling options for AstraControlCenter Resource limits. See setting complexities to understand how this settings affects others settings.
astraVersion	string	Required	1.5.2	Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version.

Setting	Type	Use	Value Example	Description
autoSupport	Undefined	Required		Indicates participation status in NetApp's pro-active support application, NetApp Active IQ. An internet connection is required (port 442) and all support data is anonymized.
autoSupport.enrolled	Boolean	Optional, but either `enrolled` or `url` fields must be selected	false (this value is the default)	Enrolled determines if you want to send anonymous data to NetApp for support purposes. The default election is `false` and indicates no support data will be sent to NetApp.
autoSupport.url	string	Optional, but either `enrolled` or `url` fields must be selected	https://support.netapp.com/asupprod/post/1.0/postAsup	URL determines where the anonymous data will be sent.
avpDeploy	Boolean	Optional	true (this is the default value)	Option that allows a user to disable deployment of a plugin operator.
crds	Undefined	Undefined		Options for how Astra Control Center should handle CRDs.
crds.externalCertManager	Boolean	Optional	True (this value is the default)	By default, Astra Control Center will install the required cert-manager CRDs. CRDs are cluster-wide objects and installing them may have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.

Setting	Type	Use	Value Example	Description
crds.externalTraefik	Boolean	Optional	True (this value is the default)	By default, Astra Control Center will install the required Traefik CRDs. CRDs are cluster-wide objects and installing them may have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.
crds.shouldUpgrade	Boolean	Optional	Undefined	Determines if CRDs should be upgraded when Astra Control Center is upgraded.
email	string	Required	admin@example.com	The username of the administrator to be added as the first user of Astra. This email address will be notified by Astra Control as events warrant.
firstName	string	Required	SRE	The first name of the administrator supporting Astra.
imageRegistry	Undefined	Optional		The container image registry that is hosting the Astra application images, Astra Control Center Operator, and Astra Control Center Helm Repository.
imageRegistry.name	string	Required if you are using imageRegistry	example.registry.com/astra	The name of the image registry. Do not prefix with protocol.

Setting	Type	Use	Value Example	Description
imageRegistry.secret	string	Required if you are using imageRegistry that requires a secret	astra-registry-cred	The name of the Kubernetes secret used to authenticate with the image registry.
ingressType	string	Optional	Generic (this is the default value)	The type of ingress Astra Control Center should be configured for. Valid values are `Generic` and `AccTraefik`. See setting complexities to understand how this settings affects others settings.
lastName	string	Required	Admin	The last name of the administrator supporting Astra.
storageClass	string	Optional (this is the default value)	ontap-gold	The storage class to be used for PVCs. If not set, the default storage class will be used.
volumeReclaimPolicy	Undefined	Optional	Retain	Reclaim policy to be set for persistent volumes.

Configuration combinations and incompatibilities

Some Astra Control Center cluster CR configuration settings greatly affect the way Astra Control Center is installed and could conflict with other settings. The content that follows describes important configuration settings and how to avoid incompatible combinations.

astraResourcesScaler

By default, Astra Control Center deploys with resource requests set for most of the components within Astra. This configuration allows the Astra Control Center software stack to perform better in environments under increased application load and scale.

However, in scenarios using smaller development or test clusters, the CR field `AstraResourcesScalar` may be set to `Off`. This disables resource requests and allows for deployment on smaller clusters.

ingressType

There are two valid values for ingressType:

- Generic
- AccTraefik

Generic (default)

When `ingressType` is set to `Generic`, Astra Control does not install any ingress resources. The assumption is that the user has a common way of securing and routing traffic through their network to applications running on Kubernetes clusters and they want to use the same mechanisms here. When the user creates an ingress to route traffic to Astra Control, the ingress needs to point to the internal traefik service on port 80. Here is an example of an Nginx ingress resource that works with the Generic ingressType setting.

```
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
```

AccTraefik

When `ingressType` is set to `AccTraefik`, Astra Control Center deploys its Traefik gateway as a Kubernetes LoadBalancer type service. Users need to provide an external Load Balancer (like MetalLB) for Astra Control Center to get an external IP.

Configure an external cert-manager

If a cert-manager already exists in your Kubernetes cluster, you need to perform some prerequisite steps so that Astra Control Center does not install its own cert-manager.

Steps

1. Confirm that you have a cert-manager installed:

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

Sample response:

```
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. Create a certificate/key pair for the astraAddress FQDN:

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

Sample response:

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

3. Create a secret with previously generated files:

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

Sample response:

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

4. Create a ClusterIssuer file that is **exactly** the following but includes the namespace location where your cert-manager pods are installed:

```
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
```

Sample response:

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

5. Verify that the ClusterIssuer has come up correctly. `Ready` must be `True` before you can proceed:

```
kubectl get ClusterIssuer
```

Sample response:

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

6. Complete the Astra Control Center installation process. There is a required configuration step for the Astra Control Center cluster YAML in which you change the CRD value to indicate that the cert-manager is externally installed. You must complete this step during installation so that Astra Control Center recognizes the external cert-manager.

Copyright information

Trademark information