# Install Astra Control Center

## Astra Control Center

NetApp
August 10, 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 to install Astra Control Center Operator and Astra Control Center in your environment. You can use this procedure to install Astra Control Center in internet-connected or air-gapped environments.

For Red Hat OpenShift environments, you can use an alternative procedure to install Astra Control Center using OpenShift OperatorHub.

**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 cluster operators are in a healthy state and available.

```
kubectl get clusteroperators
```

- 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 install its own cert-manager.

**About this task**

The Astra Control Center installation process does the following:

- Installs the Astra components into the `netapp-acc` (or custom-named) namespace.
- Creates a default account.
- Establishes a default administrative user email address and default one-time password. This user is assigned the Owner role in the system that is needed for first time login to the UI.
- Helps you determine that all Astra Control Center pods are running.
- Installs the Astra UI.

> ⓘ (Applies to the Astra Data Store Early Access Program (EAP) release only) If you intend to manage Astra Data Store using Astra Control Center and enable VMware workflows, deploy Astra Control Center only on the `pcloud` namespace and not on the `netapp-acc` namespace or a custom namespace described in the steps of this procedure.

> ⓘ Do not execute the following command during the entirety of the installation process to avoid deleting all Astra Control Center pods: `kubectl delete -f astra_control_center_operator_deploy.yaml`

> ⓘ If you are using Red Hat's Podman instead of Docker Engine, Podman commands can be used in place of Docker commands.

**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) Use the following command to 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 binaries for the plugin for different CPU architectures and operating systems. You need to know which CPU and operating system you have before you perform this task. On Linux and Mac operating systems, you can use the `uname -a` command to gather this information.

**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:

```
ls kubectl-astra/
```

2. Copy the file to the same location as the standard `kubectl` utility. In this example, the `kubectl` utility is located in the `/usr/local/bin` directory. Replace `<binary-name>` with the name of the file you need:

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

# Add the images to your local registry

1. Change to the Astra directory:

```
cd acc
```

2. 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 (for example, `acc.manifest.bundle.yaml`).

   ◦ Replace MY_REGISTRY with the URL of the Docker repository.

   ◦ Replace MY_REGISTRY_USER and MY_REGISTRY_PASSWORD with the credentials for the repository.

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

# Set up namespace and secret for registries with auth requirements

1. 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:

```
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
```

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
```

e. (Optional) If you want the cluster to be automatically managed by Astra Control Center after installation, make sure that you provide the kubeconfig as a secret within the Astra Control Center namespace you intend to deploy into using this command:

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

# Install the Astra Control Center operator

1. 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: <name_of_secret_with_creds_to_local_registry>
   ```

   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.

   d. (For installations using Astra Data Store preview) See this known issue regarding storage class
      provisioners and additional changes you will need to make to the YAML.

```yaml
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. 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
```

# Configure Astra Control Center

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

   (i) `astra_control_center_min.yaml` is the default CR and is suitable for most installations. Familiarize yourself with all CR options and their potential values to ensure you deploy Astra Control Center correctly for your environment. If additional customizations are required for your environment, you can use `astra_control_center.yaml` as an alternative CR.

   ```
   vim astra_control_center_min.yaml
   ```

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

   a. Change `[your_registry_path]` to the registry path where you pushed the images in the previous step.

   b. Change the `accountName` string to the name you want to associate with the account.

   c. Change the `astraAddress` string to the FQDN you want to use in your browser to access Astra. Do not use `http://` or `https://` in the address. Copy this FQDN for use in a later step.

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

e. Change `enrolled` for AutoSupport to `false` for sites without internet connectivity or retain `true` for connected sites.

f. If you use an external cert-manager, add the following lines to `spec`:

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

g. (Optional) 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.

h. (Optional) Change the `storageClass` value to another Trident storageClass resource if required by your installation.

i. (Optional) If you want the cluster to be automatically managed by Astra Control Center after installation and you have already created the secret containing the kubeconfig for this cluster, provide the name of the secret by adding a new field to this YAML file called `astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"`

j. Complete one of the following steps:

- **Other ingress controller (ingressType:Generic)**: This is the default action with Astra Control Center. After Astra Control Center is deployed, you will need to configure the ingress controller to expose Astra Control Center with a URL.

  The default Astra Control Center installation sets up its gateway (`service/traefik`) to be of the type `ClusterIP`. This default installation requires you to additionally set up a Kubernetes IngressController/Ingress to route traffic to it. If you want to use an ingress, see Set up ingress for load balancing.

- **Service load balancer (ingressType:AccTraefik)**: If you don't want to install an IngressController or create an Ingress resource, set `ingressType` to `AccTraefik`.

  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.

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

```
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"
```

# 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_min.yaml -n [netapp-acc or custom
   namespace]
   ```

   Sample response:

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

# Verify system status

ⓘ    If you prefer to use OpenShift, you can use comparable oc commands for verification 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-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
certificates-1a599d9f76-ab6sk               2/2      Running    0
8m35s
certificates-1a599d9f76-ab8fj               2/2      Running    0
8m52s
certificates-expiry-check-12331210--1-fc26j 1/1      Running    0
8m19s
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
keycloak-operator-5d47896894-74gq5       1/1    Running   0
8m23s
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
metrics-facade-db5c565d-5rncg            2/2    Running   0
7m39s
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
packages-94b56b6b8-jb9rk                 2/2    Running   0
10m
```

12

```
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-keycloak-0                       1/1    Running   0
7m22s
polaris-keycloak-1                       1/1    Running   0
7m19s
polaris-keycloak-2                       1/1    Running   0
7m17s
polaris-keycloak-db-0                    1/1    Running   0
7m16s
polaris-keycloak-db-1                    1/1    Running   0
7m13s
polaris-keycloak-db-2                    1/1    Running   0
7m10s
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
trident-svc-595f84dd78-zb8l6                   1/1      Running   0
8m54s
vault-controller-86c94fbf4f-krttq              1/1      Running   0
9m24s
```

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
```

> (i) `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 installation success by retrieving the `AstraControlCenter` instance installed by the Astra Control Center Operator.

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

4. In the YAML, check the `status.deploymentState` field in the response for the `Deployed` value. If deployment was unsuccessful, an error message appears instead.

5. To get the one-time password you will use when you log in to Astra Control Center, copy the `status.uuid` value. The password is `ACC-` followed by the UUID value (`ACC-[UUID]` or, in this example, `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`).

**Sample YAML Details**

```yaml
  name: astra
  namespace: netapp-acc
  resourceVersion: "104424560"
  uid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
spec:
  accountName: Example
  additionalValues: {}
  astraAddress: astra.example.com
  astraKubeConfigSecret: ""
  astraResourcesScaler: "Off"
  astraVersion: 22.08.0-18
  autoSupport:
    enrolled: true
    url: https://support.netapp.com/asupprod/post/1.0/postAsup
  avpDeploy: false
  crds: {}
  email: admin@example.com
  firstName: SRE
  imageRegistry:
    name: registry_name/astra
  ingressType: AccTraefik
  lastName: Admin
  mtls:
    certDuration: 2140h0m0s
    enabled: true
status:
  accConditionHistory:
    items:
    - astraVersion: 22.08.0-18
      condition:
        lastTransitionTime: "2022-08-05T18:03:46Z"
        message: Deploying is currently in progress.
        reason: InProgress
        status: "False"
        type: Ready
      generation: 2
      observedSpec:
        accountName: Example
        additionalValues: {}
        astraAddress: astra.example.com
        astraKubeConfigSecret: ""
        astraResourcesScaler: "Off"
        astraVersion: 22.08.0-18
        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
      lastName: Admin
      mtls:
       certDuration: 2140h0m0s
       enabled: true
    timestamp: "2022-08-05T18:03:46Z"
  - astraVersion: 22.08.0-18
    condition:
      lastTransitionTime: "2022-08-05T18:03:46Z"
      message: Deploying is currently in progress.
      reason: InProgress
      status: "True"
      type: Deploying
    generation: 2
    observedSpec:
      accountName: Example
      additionalValues: {}
      astraAddress: astra.example.com
      astraKubeConfigSecret: ""
      astraResourcesScaler: "Off"
      astraVersion: 22.08.0-18
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      avpDeploy: false
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
      lastName: Admin
      mtls:
        certDuration: 2140h0m0s
        enabled: true
    timestamp: "2022-08-05T18:03:46Z"
  - astraVersion: 22.08.0-18
    condition:
      lastTransitionTime: "2022-08-05T18:16:50Z"
      message: Post Install was successful
      observedGeneration: 2
```

```
        reason: Complete
        status: "True"
        type: PostInstallComplete
      generation: 2
      observedSpec:
        accountName: Example
        additionalValues: {}
        astraAddress: astra.example.com
        astraKubeConfigSecret: ""
        astraResourcesScaler: "Off"
        astraVersion: 22.08.0-18
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        avpDeploy: false
        crds: {}
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
        ingressType: AccTraefik
        lastName: Admin
        mtls:
          certDuration: 2140h0m0s
          enabled: true
      timestamp: "2022-08-05T18:16:50Z"
    - astraVersion: 22.08.0-18
      condition:
        lastTransitionTime: "2022-08-05T18:03:46Z"
        message: Deploying succeeded.
        reason: Complete
        status: "False"
        type: Deploying
      generation: 2
      observedSpec:
        accountName: Example
        additionalValues: {}
        astraAddress: astra.example.com
        astraKubeConfigSecret: ""
        astraResourcesScaler: "Off"
        astraVersion: 22.08.0-18
        autoSupport:
          enrolled: true
          url: https://support.netapp.com/asupprod/post/1.0/postAsup
        avpDeploy: false
        crds: {}
```

```
        email: admin@example.com
        firstName: SRE
        imageRegistry:
          name: registry_name/astra
        ingressType: AccTraefik
        lastName: Admin
        mtls:
          certDuration: 2140h0m0s
          enabled: true
      timestamp: "2022-08-05T18:16:50Z"
  - astraVersion: 22.08.0-18
    condition:
      lastTransitionTime: "2022-08-05T18:03:46Z"
      message: Astra is deployed
      reason: Complete
      status: "True"
      type: Deployed
    generation: 2
    observedSpec:
      accountName: Example
      additionalValues: {}
      astraAddress: astra.example.com
      astraKubeConfigSecret: ""
      astraResourcesScaler: "Off"
      astraVersion: 22.08.0-18
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      avpDeploy: false
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
      ingressType: AccTraefik
      lastName: Admin
      mtls:
        certDuration: 2140h0m0s
        enabled: true
    timestamp: "2022-08-05T18:16:50Z"
  - astraVersion: 22.08.0-18
    condition:
      lastTransitionTime: "2022-08-05T18:16:50Z"
      message: Astra is deployed
      reason: Complete
      status: "True"
```

```
      type: Ready
    generation: 2
    observedSpec:
      accountName: Example
      additionalValues: {}
      astraAddress: astra.example.com
      astraKubeConfigSecret: ""
      astraResourcesScaler: "Off"
      astraVersion: 22.08.0-18
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      avpDeploy: false
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
      ingressType: AccTraefik
      lastName: Admin
      mtls:
        certDuration: 2140h0m0s
        enabled: true
    timestamp: "2022-08-05T18:16:50Z"
certManager: deploy
cluster:
  type: OCP
  vendorVersion: 4.9.29
  version: v1.22.5+a36406b
conditions:
- lastTransitionTime: "2022-08-05T18:23:41Z"
  message: Astra is deployed
  reason: Complete
  status: "True"
  type: Ready
- lastTransitionTime: "2022-08-05T18:23:41Z"
  message: Deploying succeeded.
  reason: Complete
  status: "False"
  type: Deploying
- lastTransitionTime: "2022-08-05T18:23:41Z"
  message: Post Install was successful
  observedGeneration: 2
  reason: Complete
  status: "True"
  type: PostInstallComplete
```

```
      - lastTransitionTime: "2022-08-05T18:23:41Z"
        message: Astra is deployed
        reason: Complete
        status: "True"
        type: Deployed
    deploymentState: Deployed
    observedGeneration: 2
    observedSpec:
      accountName: Example
      additionalValues: {}
      astraAddress: astra.example.com
      astraKubeConfigSecret: ""
      astraResourcesScaler: "Off"
      astraVersion: 22.08.0-18
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      avpDeploy: false
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
      ingressType: AccTraefik
      lastName: Admin
      mtls:
        certDuration: 2140h0m0s
        enabled: true
    observedVersion: 22.08.0-18
    postInstall: Complete
    serviceMesh:
      type: None
    uuid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
kind: List
metadata:
 resourceVersion: ""
 selfLink: ""
```

# Set up ingress for load balancing

You can set up a Kubernetes ingress controller that manages external access to services, such as load balancing in a cluster.

This procedure explains how to set up an ingress controller (`ingressType:Generic`). This is the default action with Astra Control Center. After Astra Control Center is deployed, you will need to configure the ingress

controller to expose Astra Control Center with a URL.

> (i) If you don't want to set up an ingress controller, you can set `ingressType:AccTraefik`).
> 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.

The steps differ depending on the type of ingress controller you use:

- 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.
- You are using Kubernetes versions between and including v1.19 and v1.22.

**Steps for Istio ingress**

1. Configure Istio ingress.

   > (i) 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 `netapp-acc` (or custom-named) namespace using either the v1beta1 (deprecated in Kubernetes version less than or 1.22) or v1 resource type for either a deprecated or a new schema:

Output:

```
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
```

For the v1 new schema, follow this sample:

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

Output:

```
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. Deploy Astra Control Center as usual.

6. Check the status of the ingress:

```
kubectl get ingress -n netapp-acc
NAME    CLASS HOSTS              ADDRESS        PORTS    AGE
ingress istio astra.example.com 172.16.103.248 80, 443 1h
```

**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 either the `v1beta1` (deprecated in Kubernetes version less than or 1.22) or `v1` resource type for either a deprecated or a new schema:

a. For a `v1beta1` deprecated schema, follow this sample:

```
apiVersion: extensions/v1beta1
kind: Ingress
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. For the `v1` new schema, follow this sample:

```
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
```

**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 you used in the `astraAddress` in the `astra_control_center_min.yaml` CR when you installed Astra Control Center.

2. Accept the self-signed certificates when 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_min.yaml` CR when you installed Astra Control Center, followed by the one-time password (`ACC-[UUID]`).

> (i) 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.

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

# What's next

Complete the deployment by performing setup tasks.

# Understand Astra Control Center cluster CR options

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

| Setting | Type | Use | Value Example | Description |
|---|---|---|---|---|
| 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 |
|---------|------|-----|---------------|-------------|
| 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. |
| accountName | string | Required | Example | Astra Control Center account name. There can be only one. |
| email | string | Required | [admin@example.com](mailto: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. |
| 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. |
| astraResourcesScaler | string | Required | Default | Scaling options for AstraControlCenter Resource limits. See setting complexities to understand how this settings affects others settings. |

| Setting | Type | Use | Value Example | Description |
|---------|------|-----|---------------|-------------|
| 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. |
| 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. |
| avpDeploy | Boolean | Optional | true (this is the default value) | Option that allows a user to disable deployment of Astra Plugin for VMware vSphere operator. |
| 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. |
| imageRegistry.secret | string | Required if you are using imageRegistry | astra-registry-cred | The name of the Kubernetes secret used to authenticate with the image registry. |
| 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. |

| Setting | Type | Use | Value Example | Description |
|---|---|---|---|---|
| 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. |
| crds | Undefined | Undefined | | Options for how Astra Control Center should handle CRDs. |
| 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. |

| Setting | Type | Use | Value Example | Description |
|---------|------|-----|---------------|-------------|
| 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. |
| crds.shouldUpgrade | Boolean | Optional | Undefined | Determines if CRDs should be upgraded when Astra Control Center is upgraded. |
| mtls | | | | Options for how Astra Control Center should implement service to service mTLS in the cluster. See setting complexities to understand how this settings affects others settings |
| mtls.enabled | Boolean | Optional | true (this value is the default) | By default, Astra Control Center uses mTLS for service-to-service communication. This option should be disabled when using a service mesh to encrypt service-to-service communication instead. |

| Setting | Type | Use | Value Example | Description |
|---------|------|-----|---------------|-------------|
| mtls.certDuration | string | Optional | 2140h (this value is the default duration) | The duration of time in hours to use as a certificate lifespan when issuing service TLS certificates. This setting only works when `mtls.enabled` is set to `true`. |

## 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
```

> ⚠ When mTLS is disabled using the mtls.enabled setting in the CR, you must use `ingressType: Generic`.

**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.

**mtls**

The settings used in the CR determine how intra-application communication is secured. It is very important for the user to know ahead of time whether they will be using a service mesh or not.

* `enabled=true`: When this setting is enabled, Astra will deploy an internal service-to-service communication network that secures all traffic within the application.

> ⚠ Do not cover Astra Control Center in a service mesh while this setting is `true`.

* `enabled=false`: When this setting is disabled, Astra Control Center will not secure internal traffic and you must secure Astra namespaces independently with a service mesh.

> ⚠ When mTLS is disabled using the mtls.enabled setting in the CR, you must use `ingressType: Generic`.

!  If no service mesh is used and this setting is disabled, internal communication will be unsecure.

# Understand pod security policy restrictions

Astra Control Center supports privilege limitation through pod security policies (PSPs). Pod security policies enable you to limit what users or groups are able to run containers and what privileges those containers can have.

Some Kubernetes distributions, such as RKE2, have a default pod security policy that is too restrictive, and causes problems when installing Astra Control Center.

You can use the information and examples included here to understand the pod security policies that Astra Control Center creates, and configure pod security policies that provide the protection you need without interfering with Astra Control Center functions.

## PSPs installed by Astra Control Center

Astra Control Center creates several pod security policies during installation. Some of these are permanent, and some of them are created during certain operations and are removed once the operation is complete.

### PSPs created during installation

During Astra Control Center installation, the Astra Control Center operator installs a custom pod security policy, a Role object, and a RoleBinding object to support the deployment of Astra Control Center services in the Astra Control Center namespace.

The new policy and objects have the following attributes:

```
$ 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
```

## PSPs created during backup operations

During backup operations, Astra Control Center creates a dynamic pod security policy, a ClusterRole object, and a RoleBinding object. These support the backup process, which happens in a separate namespace.

The new policy and objects have the following attributes:

```
$ 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
```

## PSPs created during cluster management

When you manage a cluster, Astra Control Center installs the netapp-monitoring operator in the managed cluster. This operator creates a pod security policy, a ClusterRole object, and a RoleBinding object to deploy telemetry services in the Astra Control Center namespace.

The new policy and objects have the following attributes:

```
$ 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
```

## Enable network communication between namespaces

Some environments use NetworkPolicy constructs to restrict traffic between namespaces. The Astra Control Center operator, Astra Control Center, and the Astra Plugin for VMware vSphere are all in different namespaces. The services in these different namespaces need to be able to communicate with one another. To enable this communication, follow these steps.

**Steps**

1. Delete any NetworkPolicy resources that exist in the Astra Control Center namespace:

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

2. For each NetworkPolicy object that is returned by the preceding command, use the following command to delete it. Replace <OBJECT_NAME> with the name of the returned object:

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

3. Apply the following resource file to configure the acc-avp-network-policy object to allow Astra Plugin for VMware vSphere services to make requests to Astra Control Center services. Replace the information in brackets <> with information from your environment:

   ```
   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. Apply the following resource file to configure the acc-operator-network-policy object to allow the Astra Control Center operator to communicate with Astra Control Center services. Replace the information in brackets <> with information from your environment:

```yaml
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
```

## Remove resource limitations

Some environments use the ResourceQuotas and LimitRanges objects to prevent the resources in a namespace from consuming all available CPU and memory on the cluster. Astra Control Center does not set maximum limits, so it will not be in compliance with those resources. You need to remove them from the namespaces where you plan to install Astra Control Center.

You can use the following steps to retrieve and remove these quotas and limits. In these examples, the command output is shown immediately after the command.

**Steps**

1. Get the resource quotas in the netapp-acc namespace:

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

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. Delete all of the resource quotas by name:

```
$ kubectl delete resourcequota  pods-high -n netapp-acc
resourcequota "pods-high" deleted

$ kubectl delete resourcequota  pods-low -n netapp-acc
resourcequota "pods-low" deleted

$ kubectl delete resourcequota  pods-medium -n netapp-acc
resourcequota "pods-medium" deleted
```

3. Get the limit ranges in the netapp-acc namespace:

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

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

4. Delete the limit ranges by name:

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

# 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.