Set up Astra Control Center

Astra Control Center

NetApp
August 10, 2022

Table of Contents

Set up Astra Control Center

Astra Control Center supports and monitors ONTAP and Astra Data Store as the storage backend. After you install Astra Control Center, log in to the UI, and change your password, you'll want to set up a license, add clusters, manage storage, and add buckets.

Tasks

- Add a license for Astra Control Center
- Add cluster
- Add a storage backend
- Add a bucket

Add a license for Astra Control Center

You can add a new license using the UI or API to gain full Astra Control Center functionality. Without a license, your usage of Astra Control Center is limited to managing users and adding new clusters.

For more information on how licenses are calculated, see Licensing.

To update an existing evaluation or full license, see Update an existing license.

Astra Control Center licenses measure CPU resources using Kubernetes CPU units. The license needs to account for the CPU resources assigned to the worker nodes of all the managed Kubernetes clusters. Before you add a license, you need to obtain the license file (NLF) from the NetApp Support Site.

You can also try Astra Control Center with an evaluation license, which lets you use Astra Control Center for 90 days from the date you download the license. You can sign up for a free trial by registering here.

If your installation grows to exceed the licensed number of CPU units, Astra Control Center prevents you from managing new applications. An alert is displayed when capacity is exceeded.

What you'll need

When you downloaded Astra Control Center from the NetApp Support Site, you also downloaded the NetApp license file (NLF). Ensure you have access to this license file.

Steps

1. Log in to the Astra Control Center UI.
2. Select **Account** > **License**.
3. Select **Add License**.
4. Browse to the license file (NLF) that you downloaded.
5. Select **Add License**.

The **Account** > **License** page displays the license information, expiration date, license serial number, account ID, and CPU units used.

If you have an evaluation license, be sure you store your account ID to avoid data loss in the event of Astra Control Center failure if you are not sending ASUPs.

Add cluster

To begin managing your apps, add a Kubernetes cluster and manage it as a compute resource. You have to add a cluster for Astra Control Center to discover your Kubernetes applications. For Astra Data Store, you want to add the Kubernetes app cluster that contains applications that are using volumes provisioned by Astra Data Store.

We recommend that Astra Control Center manage the cluster it is deployed on first before you add other clusters to Astra Control Center to manage. Having the initial cluster under management is necessary to send Kubemetrics data and cluster-associated data for metrics and troubleshooting. You can use the **Add Cluster** feature to manage a cluster with Astra Control Center.

When Astra Control manages a cluster, it keeps track of the cluster's default storage class. If you change the storage class using `kubectl` commands, Astra Control reverts the change. To change the default storage class in a cluster managed by Astra Control, use one of the following methods:

- Use the Astra Control API `PUT /managedClusters` endpoint, and assign a different default storage class with the `DefaultStorageClass` parameter.
- Use the Astra Control web UI to assign a different default storage class. See Change the default storage class.

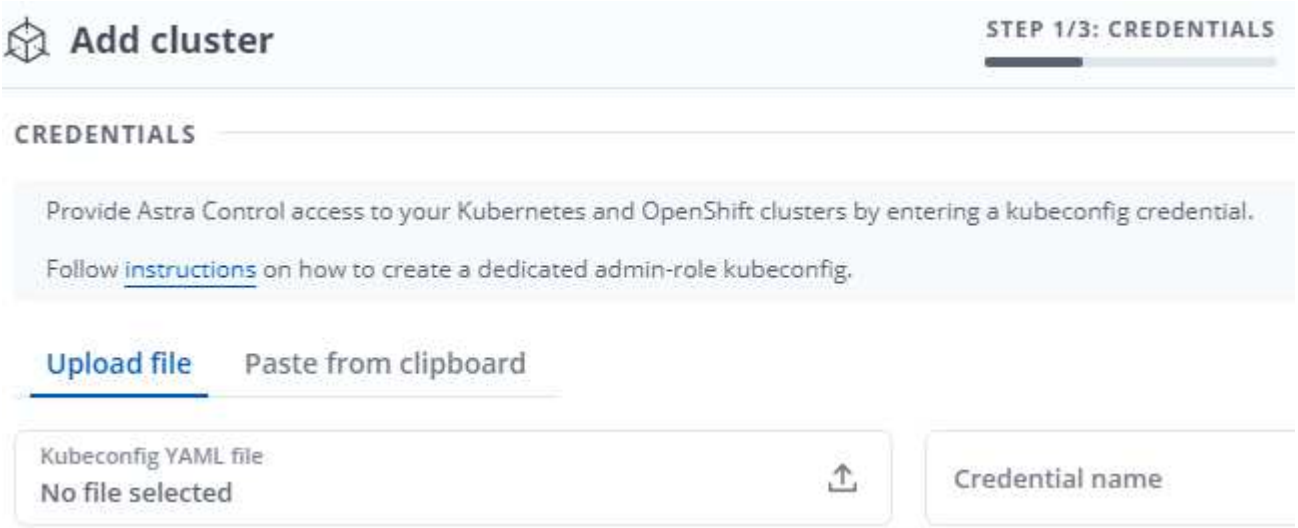
What you'll need

- Before you add a cluster, review and perform the necessary prerequisite tasks.

Steps

1. From the **Dashboard** in the Astra Control Center UI, select **Add** in the Clusters section.
2. In the **Add Cluster** window that opens, upload a `kubeconfig.yaml` file or paste the contents of a `kubeconfig.yaml` file.

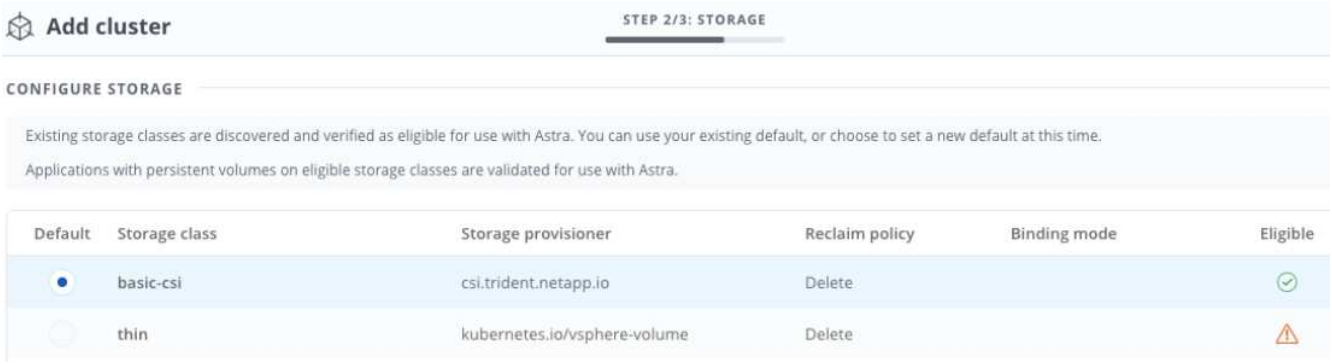
   The `kubeconfig.yaml` file should include **only the cluster credential for one cluster**.



If you create your own `kubeconfig` file, you should define only **one** context element in it. See Kubernetes documentation for information about creating `kubeconfig` files.

3. Provide a credential name. By default, the credential name is auto-populated as the name of the cluster.
4. Select **Configure storage**.
5. Select the storage class to be used for this Kubernetes cluster, and select **Review**.

You should select a Trident storage class backed by ONTAP storage or Astra Data Store.



6. Review the information, and if everything looks good, select **Add cluster**.

Result

The cluster enters the **Discovering** status and then changes to **Running**. You have successfully added a Kubernetes cluster and are now managing it in Astra Control Center.

After you add a cluster to be managed in Astra Control Center, it might take a few minutes to deploy the monitoring operator. Until then, the Notification icon turns red and logs a **Monitoring Agent Status Check Failed** event. You can ignore this, because the issue resolves when Astra Control Center obtains the correct status. If the issue does not resolve in a few minutes, go to the cluster, and run `oc get pods -n netapp-monitoring` as the starting point. You will need to look into the monitoring operator logs to debug the problem.

Add a storage backend

You can add a storage backend so that Astra Control can manage its resources. You can deploy a storage backend on a managed cluster or use an existing storage backend.

Managing storage clusters in Astra Control as a storage backend enables you to get linkages between persistent volumes (PVs) and the storage backend as well as additional storage metrics.

What you'll need for existing Astra Data Store deployments

- You have added your Kubernetes app cluster and the underlying compute cluster.

After you add your Kubernetes app cluster for Astra Data Store and it is managed by Astra Control, the cluster appears as `unmanaged` in the list of discovered backends. You must next add the compute cluster that contains Astra Data Store and underlies the Kubernetes app cluster. You can do this from **Backends** in the UI. Select the Actions menu for the cluster, select `Manage`, and add the cluster. After the cluster state of `unmanaged` changes to the name of the Kubernetes cluster, you can proceed with adding a backend.

What you'll need for new Astra Data Store deployments

- You have uploaded the version of the installation bundle you intend to deploy to a location that is accessible to Astra Control.
- You have added the Kubernetes cluster that you intend to use for deployment.
- You have uploaded the Astra Data Store license for your deployment to a location that is accessible to Astra Control.

Options

- Deploy storage resources
- Use an existing storage backend

Deploy storage resources

You can deploy a new Astra Data Store and manage the associated storage backend.

Steps

1. Navigate from the Dashboard or the Backends menu:
   - From **Dashboard**: From the Resource Summary, select a link from the Storage Backends pane and select **Add** from the Backends section.
   - From **Backends**:
     a. In the left navigation area, select **Backends**.
     b. Select **Add**.
2. Select the **Astra Data Store** deployment option within the **Deploy** tab.
3. Select the Astra Data Store package to deploy:
   a. Enter a name for the Astra Data Store application.
   b. Choose the version of Astra Data Store you want to deploy.

   If you have not yet uploaded the version you intend to deploy, you can use the **Add package** option or exit the wizard and use package management to upload the installation bundle.

4. Select an Astra Data Store license that you have previously uploaded or use the **Add license** option to upload a license to use with the application.

   Astra Data Store licenses with full permissions are associated with your Kubernetes cluster, and these associated clusters should appear automatically. If there is no managed cluster, you can select the **Add a Cluster** option to add one to Astra Control management.
   For Astra Data Store licenses, if no association has been made between the license and cluster, you can define this association on the next page of the wizard.

5. If you have not added a Kubernetes cluster to Astra Control management, you need to do so from the **Kubernetes cluster** page. Select an existing cluster from the list or select **add the underlying cluster** to add a cluster to Astra Control management.
6. Select a template size for the Kubernetes cluster that will provide resources for Astra Data Store. You can choose one of the following:
   - If you choose `Recommended Kubernetes worker node requirements`, select a template from large to small based on what your license allows.
   - If you choose `Custom Kubernetes worker node requirements`, select the number of cores and total memory you want for each cluster node. You can also show the eligible number of nodes in the cluster that meet your selection criteria for cores and memory.

When picking a template, select larger nodes with more memory and cores for larger workloads or a greater number of nodes for smaller workloads. You should select a template based on what your license allows. Each recommended template option suggests the number of eligible nodes that satisfy the template pattern for memory and cores and capacity for each node.

7. Configure the nodes:
   a. Add a node label to identify the pool of worker nodes that supports this Astra Data Store cluster.

The label must be added to each individual node in the cluster that will be used for Astra Data Store deployment prior to the start of deployment or deployment will fail.

   b. Configure the capacity (GiB) per node manually or select the maximum node capacity allowed.
   c. Configure a maximum number of nodes allowed in the cluster or allow the maximum number of nodes on the cluster.
8. (Astra Data Store full licenses only) Enter the key of the label you want to use for Protection Domains.

Create at least three unique labels for the key for each node. For example, if your key is `astra.datastore.protection.domain`, you might create the following labels: `astra.datastore.protection.domain=domain1`,`astra.datastore.protection.domain=domain2`, and `astra.datastore.protection.domain=domain3`.

9. Configure the management network:
   a. Enter a management IP address for Astra Data Store internal management that is on the same subnet as worker node IP addresses.
   b. Choose to use the same NIC for both management and data networks or configure them separately.
   c. Enter data network IP address pool, subnet mask and gateway for storage access.
10. Review the configuration and select **Deploy** to begin installation.

Result

After a successful installation, the backend appears in `available` state in the backends list along with active performance information.

You might need to refresh the page for the backend to appear.

Use an existing storage backend

You can bring a discovered ONTAP or Astra Data Store storage backend into Astra Control Center management.

Steps

1. Navigate from the Dashboard or the Backends menu:
   - From **Dashboard**: From the Resource Summary, select a link from the Storage Backends pane and select **Add** from the Backends section.
   - From **Backends**:
     a. In the left navigation area, select **Backends**.
     b. Select **Manage** on a discovered backend from the managed cluster or select **Add** to manage an additional existing backend.
2. Select the **Use existing** tab.
3. Do one of the following depending on your backend type:
   - **Astra Data Store**:
     a. Select **Astra Data Store**.
     b. Select the managed compute cluster and select **Next**.
     c. Confirm the backend details and select **Add storage backend**.
   - **ONTAP**:
     a. Select **ONTAP**.
     b. Enter the ONTAP admin credentials and select **Review**.
     c. Confirm the backend details and select **Add storage backend**.

Result

The backend appears in `available` state in the list with summary information.

You might need to refresh the page for the backend to appear.

Add a bucket

Adding object store bucket providers is essential if you want to back up your applications and persistent storage or if you want to clone applications across clusters. Astra Control stores those backups or clones in the object store buckets that you define.

When you add a bucket, Astra Control marks one bucket as the default bucket indicator. The first bucket that you create becomes the default bucket.

You don’t need a bucket if you are cloning your application configuration and persistent storage to the same cluster.

Use any of the following bucket types:

- NetApp ONTAP S3
- NetApp StorageGRID S3

- Generic S3
- Microsoft Azure

> Although Astra Control Center supports Amazon S3 as a Generic S3 bucket provider, Astra Control Center might not support all object store vendors that claim Amazon's S3 support.

For instructions on how to add buckets using the Astra Control API, see Astra Automation and API information.

Steps

1. In the left navigation area, select **Buckets**.
   a. Select **Add**.
   b. Select the bucket type.

      > When you add a bucket, select the correct bucket provider and provide the right credentials for that provider. For example, the UI accepts NetApp ONTAP S3 as the type and accepts StorageGRID credentials; however, this will cause all future app backups and restores using this bucket to fail.

   c. Create a new bucket name or enter an existing bucket name and optional description.

      > The bucket name and description appear as a backup location that you can choose later when you're creating a backup. The name also appears during protection policy configuration.

   d. Enter the name or IP address of the S3 endpoint.
   e. If you want this bucket to be the default bucket for all backups, check the `Make this bucket the default bucket for this private cloud` option.

      > This option does not appear for the first bucket you create.

   f. Continue by adding credential information.

Add S3 access credentials

Add S3 access credentials at any time.

Steps

1. From the Buckets dialog, select either the **Add** or **Use existing** tab.
   a. Enter a name for the credential that distinguishes it from other credentials in Astra Control.
   b. Enter the access ID and secret key by pasting the contents from your clipboard.

Change the default storage class

You can change the default storage class for a cluster.

Steps

1. In the Astra Control Center web UI, select **Clusters**.

2. On the **Clusters** page, select the cluster that you want to change.
3. Select the **Storage** tab.
4. Select the **Storage classes** category.
5. Select the **Actions** menu for the storage class that you want to set as default.
6. Select **Set as default**.

What's next?

Now that you've logged in and added clusters to Astra Control Center, you're ready to start using Astra Control Center's application data management features.

- Manage users
- Start managing apps
- Protect apps
- Clone apps
- Manage notifications
- Connect to Cloud Insights
- Add a custom TLS certificate

Find more information

- Use the Astra Control API
- Known issues

Prerequisites for adding a cluster

You should ensure that the prerequisite conditions are met before you add a cluster. You should also run the eligibility checks to ensure that your cluster is ready to be added to Astra Control Center.

What you'll need before you add a cluster

Ensure that your cluster meets the requirements outlined in Application cluster requirements.

> If you plan to add a second OpenShift 4.6, 4.7, or 4.8 cluster as a managed compute resource, you should ensure that the Astra Trident Volume Snapshot feature is enabled. See the official Astra Trident instructions to enable and test Volume Snapshots with Astra Trident.

- Astra Trident StorageClasses configured with a supported storage backend (required for any type of cluster)
- The superuser and user ID set on the backing ONTAP system to back up and restore apps with Astra Control Center. Run the following command in the ONTAP command line:
  `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser sysm --anon 65534`
- An Astra Trident `volumesnapshotclass` object that has been defined by an administrator. See the Astra Trident instructions to enable and test Volume Snapshots with Astra Trident.

- Ensure that you have only a single default storage class defined for your Kubernetes cluster.

Run eligibility checks

Run the following eligibility checks to ensure that your cluster is ready to be added to Astra Control Center.

Steps

1. Check the Trident version.

   ```
   kubectl get tridentversions -n trident
   ```

   If Trident exists, you see output similar to the following:

   ```
   NAME      VERSION
   trident   21.04.0
   ```

   If Trident does not exist, you see output similar to the following:

   ```
   error: the server doesn't have a resource type "tridentversions"
   ```

   If Trident is not installed or the installed version is not the latest, you need to install the latest version of Trident before proceeding. See the Trident documentation for instructions.

2. Check if the storage classes are using the supported Trident drivers. The provisioner name should be `csi.trident.netapp.io`. See the following example:

   ```
   kubectl get sc
   NAME                    PROVISIONER                    RECLAIMPOLICY
   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
   ontap-gold (default)    csi.trident.netapp.io          Delete
   Immediate           true                   5d23h
   thin                    kubernetes.io/vsphere-volume   Delete
   Immediate           false                  6d
   ```

Create an admin-role kubeconfig

Ensure that you have the following on your machine before you do the steps:

- `kubectl` v1.19 or later installed
- An active kubeconfig with cluster admin rights for the active context

Steps

1. Create a service account as follows:

a. Create a service account file called `astracontrol-service-account.yaml`.

   Adjust the name and namespace as needed. If changes are made here, you should apply the same changes in the following steps.

```
<strong>astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

b. Apply the service account:

```
kubectl apply -f astracontrol-service-account.yaml
```

2. (Optional) If your cluster uses a restrictive pod security policy that doesn't allow privileged pod creation or allow processes within the pod containers to run as the root user, create a custom pod security policy for the cluster that enables Astra Control to create and manage pods. For instructions, see Create a custom pod security policy.
3. Grant cluster admin permissions as follows:
   a. Create a `ClusterRoleBinding` file called `astracontrol-clusterrolebinding.yaml`.

      Adjust any names and namespaces modified when creating the service account as needed.

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

b. Apply the cluster role binding:

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

4. List the service account secrets, replacing `<context>` with the correct context for your installation:

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

The end of the output should look similar to the following:

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-vhz87"},
{ "name": "astracontrol-service-account-token-r59kr"}
]
```

The indices for each element in the `secrets` array begin with 0. In the above example, the index for `astracontrol-service-account-dockercfg-vhz87` would be 0 and the index for `astracontrol-service-account-token-r59kr` would be 1. In your output, make note of the index for the service account name that has the word "token" in it.

5. Generate the kubeconfig as follows:

a. Create a `create-kubeconfig.sh` file. Replace `TOKEN_INDEX` in the beginning of the following script with the correct value.

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
```

```
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token
-user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. Source the commands to apply them to your Kubernetes cluster.

```
source create-kubeconfig.sh
```

6. (**Optional**) Rename the kubeconfig to a meaningful name for your cluster. Protect your cluster credential.

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

What's next?

Now that you've verified that the prerequisites are met, you're ready to add a cluster.

Find more information

- Trident documentation
- Use the Astra Control API

Add a custom TLS certificate

You can remove the existing self-signed TLS certificate and replace it with a TLS certificate signed by a Certificate Authority (CA).

What you'll need

- Kubernetes cluster with Astra Control Center installed
- Administrative access to a command shell on the cluster to run `kubectl` commands
- Private key and certificate files from the CA

Remove the self-signed certificate

Remove the existing self-signed TLS certificate.

1. Using SSH, log in to the Kubernetes cluster that hosts Astra Control Center as an administrative user.
2. Find the TLS secret associated with the current certificate using the following command, replacing `<ACC-deployment-namespace>` with the Astra Control Center deployment namespace:

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

3. Delete the currently installed secret and certificate using the following commands:

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

Add a new certificate

Add a new TLS certificate that is signed by a CA.

1. Use the following command to create the new TLS secret with the private key and certificate files from the CA, replacing the arguments in brackets <> with the appropriate information:

```
kubectl create secret tls <secret-name> --key <private-key-filename>
--cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. Use the following command and example to edit the cluster Custom Resource Definition (CRD) file and change the `spec.selfSigned` value to `spec.ca.secretName` to refer to the TLS secret you created earlier:

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

#spec:
#  selfSigned: {}

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

3. Use the following command and example output to validate that the changes are correct and the cluster is ready to validate certificates, replacing `<ACC-deployment-namespace>` with the Astra Control Center deployment namespace:

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

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

4. Create the `certificate.yaml` file using the following example, replacing the placeholder values in brackets <> with appropriate information:

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

5. Create the certificate using the following command:

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

6. Using the following command and example output, validate that the certificate has been created correctly and with the arguments you specified during creation (such as name, duration, renewal deadline, and DNS names).

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

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

7. Edit the ingress CRD TLS option to point to your new certificate secret using the following command and example, replacing the placeholder values in brackets <> with appropriate information:

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

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

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

8. Using a web browser, browse to the deployment IP address of Astra Control Center.
9. Verify that the certificate details match the details of the certificate you installed.
10. Export the certificate and import the result into the certificate manager in your web browser.

Create a custom pod security policy

Astra Control needs to create and manage Kubernetes pods on the clusters it manages. If your cluster uses a restrictive pod security policy that doesn't allow privileged pod creation or allow processes within the pod containers to run as the root user, you need to create a less restrictive pod security policy to enable Astra Control to create and manage these pods.

Steps

1. Create a pod security policy for the cluster that is less restrictive than the default, and save it in a file. For example:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: astracontrol
  annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
```

2. Create a new role for the pod security policy.

```
kubectl-admin create role psp:astracontrol \
    --verb=use \
    --resource=podsecuritypolicy \
    --resource-name=astracontrol
```

3. Bind the new role to the service account.

```
kubectl-admin create rolebinding default:psp:astracontrol \
    --role=psp:astracontrol \
    --serviceaccount=astracontrol-service-account:default
```