NetApp

Installation overview

Astra Control Center

NetApp
February 12, 2024

Table of Contents

Installation overview

Choose and complete one of the following Astra Control Center installation procedures:

- Install Astra Control Center using the standard process
- (If you use Red Hat OpenShift) Install Astra Control Center using OpenShift OperatorHub
- Install Astra Control Center with a Cloud Volumes ONTAP storage backend

Depending on your environment, there might be additional configuration needed after you install Astra Control Center:

- Configure Astra Control Center after installation

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.

For a demonstration of the Astra Control Center installation process, see this video.

Before you begin

- 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. Refer to 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.

> Deploy Astra Control Center in a third fault domain or secondary site. This is recommended for app replication and seamless disaster recovery.

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 extract Astra Control Center
- 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 extract Astra Control Center

1. Go to the Astra Control Center downloads page on the NetApp Support Site.

2. Download the bundle containing Astra Control Center (`astra-control-center-[version].tar.gz`).

3. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (`astra-control-center-certs-[version].tar.gz`) to verify the signature of the bundle:

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

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

   The output will show `Verified OK` after successful verification.

4. Extract the images from the Astra Control Center bundle:

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

Install the NetApp Astra kubectl plugin

You can use the NetApp Astra kubectl command line plugin to push images to a local Docker repository.

Before you begin

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.

If you already have the plugin installed from a previous installation, make sure you have the latest version before completing these steps.

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:

   (i) 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 root directory of the tarball. You should see this file and directory:

```
acc.manifest.bundle.yaml
acc/
```

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

- Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.bundle.yaml`).
- Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, "https://<docker-registry>".
- 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. Change to the root directory of the tarball. You should see this file and directory:

```
acc.manifest.bundle.yaml
acc/
```

b. Log in to your registry:

```
podman login <YOUR_REGISTRY>
```

c. Prepare and run one of the following scripts that is customized for the version of Podman you use. Substitute <MY_FULL_REGISTRY_PATH> with the URL of your repository that includes any sub-directories.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

> ⓘ The image path the script creates should resemble the following, depending on your
> registry configuration:

```
https://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

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]
```

> (i) 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:

   > (i) The placeholder `your_registry_path` should match the location of the images that you uploaded earlier (for example, `[Registry_URL]/netapp/astra/astracc/23.04.2-7`).

   ```
   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
   ```

   > (i) 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
```

(i)   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
```

```yaml
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:23.04.36
        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, support, registry, and other necessary configurations:

```
vim astra_control_center.yaml
```

> ⓘ An annotated sample YAML follows these steps.

2. Modify or confirm the following settings:

`accountName`

Setting	Guidance	Type	Example
accountName	Change the `accountName` string to the name you want to associate with the Astra Control Center account. There can be only one accountName.	string	Example

`astraVersion`

Setting	Guidance	Type	Example
astraVersion	The version of Astra Control Center to deploy. No action is needed for this setting as the value will be pre-populated.	string	23.04.2-7

`astraAddress`

Setting	Guidance	Type	Example
`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 address defines how Astra Control Center will be found in your data center and is the same FQDN or IP address you provisioned from your load balancer when you completed Astra Control Center requirements. NOTE: Do not use `http://` or `https://` in the address. Copy this FQDN for use in a later step.	string	`astra.example.com`

`autoSupport`

Your selections in this section determine whether you will participate in NetApp's pro-active support application, NetApp Active IQ, and where data is sent. An internet connection is required (port 442), and all support data is anonymized.

Setting	Use	Guidance	Type	Example
`autoSupport.enrolled`	Either `enrolled` or `url` fields must be selected	Change `enrolled` for AutoSupport to `false` for sites without internet connectivity or retain `true` for connected sites. A setting of `true` enables anonymous data to be sent to NetApp for support purposes. The default election is `false` and indicates no support data will be sent to NetApp.	Boolean	`false` (this value is the default)
`autoSupport.url`	Either `enrolled` or `url` fields must be selected	This URL determines where the anonymous data will be sent.	string	https://support.netapp.com/asupprod/post/1.0/postAsup

`email`

Setting	Guidance	Type	Example
`email`	Change the `email` string to the default initial administrator address. Copy this email address for use in a later step. This email address will be used as the username for the initial account to log in to the UI and will be notified of events in Astra Control.	string	admin@example.com

`firstName`

Setting	Guidance	Type	Example
`firstName`	The first name of the default initial administrator associated with the Astra account. The name used here will be visible in a heading in the UI after your first login.	string	`SRE`

`LastName`

Setting	Guidance	Type	Example
`lastName`	The last name of the default initial administrator associated with the Astra account. The name used here will be visible in a heading in the UI after your first login.	string	`Admin`

```
imageRegistry
```

Your selections in this section define the container image registry that is hosting the Astra application images, Astra Control Center Operator, and Astra Control Center Helm repository.

| Setting	Use	Guidance	Type	Example
`imageRegistry.name`	Required	The name of the image registry where you pushed the images in the previous step. Do not use `http://` or `https://` in the registry name.	string	`example.registry.com/astra`
`imageRegistry.secret`	Required if the string you entered for `imageRegistry.name' requires a secret.` `IMPORTANT: If you are using a registry that does not require authorization, you must delete this `secret` line within imageRegistry or the installation will fail.	The name of the Kubernetes secret used to authenticate with the image registry.	string	`astra-registry-cred`

`storageClass`

Setting	Guidance	Type	Example
`storageClass`	Change the `storageClass` value from `ontap-gold` to another Astra Trident storageClass resource as required by your installation. Run the command `kubectl get sc` to determine your existing configured storage classes. One of the Astra Trident-based storage classes must be entered in the manifest file (`astra-control-center-<version>.manifest`) and will be used for Astra PVs. If it is not set, the default storage class will be used. NOTE: If a default storage class is configured, ensure that it is the only storage class that has the default annotation.	string	`ontap-gold`

`volumeReclaimPolicy`

Setting	Guidance	Type	Options
`volumeReclaimPolicy`	This sets the reclaim policy for Astra's PVs. Setting this policy to `Retain` retains persistent volumes after Astra is deleted. Setting this policy to `Delete` deletes persistent volumes after astra is deleted. If this value is not set, the PVs are retained.	string	• `Retain` (This is the default value) • `Delete`

ingressType

Setting	Guidance	Type	Options
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	string	• Generic (this is the default value) • AccTraefik

`scaleSize`

Setting	Guidance	Type	Options
`scaleSize`	By default, Astra will use High Availability (HA) `scaleSize` of `Medium`, which deploys most services in HA and deploys multiple replicas for redundancy. With `scaleSize` as `Small`, Astra will reduce the number of replicas for all services except for essential services to reduce consumption. TIP: `Medium` deployments consist of around 100 pods (not including transient workloads. 100 pods is based on a three master node and three worker node configuration). Be aware of per-pod network limit constraints that might be an issue in your environment, especially when considering disaster recovery scenarios.	string	`Small` `Medium` (This is the default value)

`astraResourcesScaler`

Setting	Guidance	Type	Options
`astraResourcesScaler`	Scaling options for AstraControlCenter Resource limits. 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.	string	• `Default` (This is the default value) • `Off`

`additionalValues`

- For Astral Control Center and Cloud Insights communication, TLS certificate verification is disabled by default. You can enable TLS certification verification for communication between Cloud Insights and both the Astra Control Center host cluster and managed cluster by adding the following section in `additionalValues`.

```
additionalValues:
  netapp-monitoring-operator:
    config:
      ciSkipTlsVerify: false
  cloud-insights-service:
    config:
      ciSkipTlsVerify: false
  telemetry-service:
    config:
      ciSkipTlsVerify: false
```

```
crds
```

Your selections in this section determine how Astra Control Center should handle CRDs.

Setting	Guidance	Type	Example
`crds.externalCert Manager`	If you use an external cert manager, change `externalCertManag er` to `true`. The default `false` causes Astra Control Center to install its own cert manager CRDs during installation. CRDs are cluster-wide objects and installing them might 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.	Boolean	`False` (this value is the default)
`crds.externalTrae fik`	By default, Astra Control Center will install required Traefik CRDs. CRDs are cluster-wide objects and installing them might 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.	Boolean	`False` (this value is the default)

(i) Be sure that you have selected the correct storage class and ingress type for your configuration before completing 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"
  scaleSize: "Medium"
  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
```

> ⓘ The Astra Control Center operator will run an automatic check for environment requirements. Missing requirements can cause your installation to fail or Astra Control Center to not operate properly. See the next section to check for warning messages related to the automatic system check.

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 the installation process did not produce warnings messages related to the validation checks:

```
kubectl get acc [astra or custom Astra Control Center CR name] -n
[netapp-acc or custom namespace] -o yaml
```

> ⓘ Additional warning messages are also reported in the Astra Control Center operator logs.

2. Correct any issues with your environment that were reported by the automated requirements checks.

> ⓘ You can correct issues by ensuring that your environment meets the requirements for Astra Control Center.

3. 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-6cc7696d8f-pmhm8             1/1     Running     0
9h
activity-597fb656dc-5rd4l                  1/1     Running     0
9h
activity-597fb656dc-mqmcw                  1/1     Running     0
9h
api-token-authentication-62f84             1/1     Running     0
9h
api-token-authentication-68nlf             1/1     Running     0
9h
api-token-authentication-ztgrm             1/1     Running     0
9h
asup-669d4ddbc4-fnmwp                      1/1     Running     1
(9h ago)     9h
authentication-78789d7549-lk686            1/1     Running     0
9h
bucketservice-65c7d95496-24x7l             1/1     Running     3
(9h ago)     9h
cert-manager-c9f9fbf9f-k8zq2               1/1     Running     0
9h
cert-manager-c9f9fbf9f-qjlzm               1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-b5qll    1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-p5whs    1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-4722b      1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-86kv5      1/1     Running     0
9h
certificates-59d9f6f4bd-2j899              1/1     Running     0
9h
certificates-59d9f6f4bd-9d9k6              1/1     Running     0
9h
certificates-expiry-check-28011180--1-8lkxz  0/1   Completed   0
9h
cloud-extension-5c9c9958f8-jdhrp           1/1     Running     0
9h
cloud-insights-service-5cdd5f7f-pp8r5      1/1     Running     0
9h
composite-compute-66585789f4-hxn5w         1/1     Running     0
9h
```

```
composite-volume-68649f68fd-tb7p4          1/1     Running      0
9h
credentials-dfc844c57-jsx92                1/1     Running      0
9h
credentials-dfc844c57-xw26s                1/1     Running      0
9h
entitlement-7b47769b87-4jb6c               1/1     Running      0
9h
features-854d8444cc-c24b7                   1/1     Running      0
9h
features-854d8444cc-dv6sm                   1/1     Running      0
9h
fluent-bit-ds-9tlv4                         1/1     Running      0
9h
fluent-bit-ds-bpkcb                         1/1     Running      0
9h
fluent-bit-ds-cxmwx                         1/1     Running      0
9h
fluent-bit-ds-jgnhc                         1/1     Running      0
9h
fluent-bit-ds-vtr6k                         1/1     Running      0
9h
fluent-bit-ds-vxqd5                         1/1     Running      0
9h
graphql-server-7d4b9d44d5-zdbf5            1/1     Running      0
9h
identity-6655c48769-4pwk8                   1/1     Running      0
9h
influxdb2-0                                 1/1     Running      0
9h
keycloak-operator-55479d6fc6-slvmt         1/1     Running      0
9h
krakend-f487cb465-78679                     1/1     Running      0
9h
krakend-f487cb465-rjsxx                     1/1     Running      0
9h
license-64cbc7cd9c-qxsr8                    1/1     Running      0
9h
login-ui-5db89b5589-ndb96                   1/1     Running      0
9h
loki-0                                      1/1     Running      0
9h
metrics-facade-8446f64c94-x8h7b            1/1     Running      0
9h
monitoring-operator-6b44586965-pvcl4       2/2     Running      0
9h
```

```
nats-0                              1/1     Running    0
9h
nats-1                              1/1     Running    0
9h
nats-2                              1/1     Running    0
9h
nautilus-85754d87d7-756qb           1/1     Running    0
9h
nautilus-85754d87d7-q8j7d           1/1     Running    0
9h
openapi-5f9cc76544-7fnjm            1/1     Running    0
9h
openapi-5f9cc76544-vzr7b            1/1     Running    0
9h
packages-5db49f8b5-lrzhd            1/1     Running    0
9h
polaris-consul-consul-server-0      1/1     Running    0
9h
polaris-consul-consul-server-1      1/1     Running    0
9h
polaris-consul-consul-server-2      1/1     Running    0
9h
polaris-keycloak-0                  1/1     Running    2
(9h ago)    9h
polaris-keycloak-1                  1/1     Running    0
9h
polaris-keycloak-2                  1/1     Running    0
9h
polaris-keycloak-db-0               1/1     Running    0
9h
polaris-keycloak-db-1               1/1     Running    0
9h
polaris-keycloak-db-2               1/1     Running    0
9h
polaris-mongodb-0                   1/1     Running    0
9h
polaris-mongodb-1                   1/1     Running    0
9h
polaris-mongodb-2                   1/1     Running    0
9h
polaris-ui-66fb99479-qp9gq          1/1     Running    0
9h
polaris-vault-0                     1/1     Running    0
9h
polaris-vault-1                     1/1     Running    0
9h
```

```
polaris-vault-2                                  1/1    Running     0
9h
public-metrics-76fbf9594d-zmxzw                  1/1    Running     0
9h
storage-backend-metrics-7d7fbc9cb9-lmd25         1/1    Running     0
9h
storage-provider-5bdd456c4b-2fftc                1/1    Running     0
9h
task-service-87575df85-dnn2q                     1/1    Running     3
(9h ago)    9h
task-service-task-purge-28011720--1-q6w4r        0/1    Completed   0
28m
task-service-task-purge-28011735--1-vk6pd        1/1    Running     0
13m
telegraf-ds-2r2kw                                1/1    Running     0
9h
telegraf-ds-6s9d5                                1/1    Running     0
9h
telegraf-ds-96jl7                                1/1    Running     0
9h
telegraf-ds-hbp84                                1/1    Running     0
9h
telegraf-ds-plwzv                                1/1    Running     0
9h
telegraf-ds-sr22c                                1/1    Running     0
9h
telegraf-rs-4sbg8                                1/1    Running     0
9h
telemetry-service-fb9559f7b-mk9l7                1/1    Running     3
(9h ago)    9h
tenancy-559bbc6b48-5msgg                         1/1    Running     0
9h
traefik-d997b8877-7xpf4                          1/1    Running     0
9h
traefik-d997b8877-9xv96                          1/1    Running     0
9h
trident-svc-585c97548c-d25z5                     1/1    Running     0
9h
vault-controller-88484b454-2d6sr                 1/1    Running     0
9h
vault-controller-88484b454-fc5cz                 1/1    Running     0
9h
vault-controller-88484b454-jktld                 1/1    Running     0
9h
```

4. (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
```

> ℹ️ `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.

5. 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  23.04.2-7   10.111.111.111
True
```

> ℹ️ 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

Before you begin

- 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 address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    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 or custom namespace]
```

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):

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC address>
    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 (including the `https://` prefix) you used in the `astraAddress` in the `astra_control_center.yaml` CR when you installed Astra Control Center.

2. Accept the self-signed certificates if prompted.

   > ℹ️ 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]`).

   > ℹ️ 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.

   > ℹ️ 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.

Options

- 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
```

- To check the output of the Astra Control Center CR:

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

What's next

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

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.

Install Astra Control Center using OpenShift OperatorHub

If you use Red Hat OpenShift, you can install Astra Control Center using the Red Hat certified operator. Use this procedure to install Astra Control Center from the Red Hat Ecosystem Catalog or using the Red Hat OpenShift Container Platform.

After you complete this procedure, you must return to the installation procedure to complete the remaining steps to verify installation success and log on.

Before you begin

- **Environmental prerequisites met**: Before you begin installation, prepare your environment for Astra Control Center deployment.

- **Healthy cluster operators and API services**:

  ◦ From your OpenShift cluster, ensure all cluster operators are in a healthy state:

  ```
  oc get clusteroperators
  ```

  ◦ From your OpenShift cluster, ensure all API services are in a healthy state:

  ```
  oc get apiservices
  ```

- **FQDN address**: Obtain an FQDN address for Astra Control Center in your data center.

- **Openshift Permissions**: Obtain the necessary permissions and access to the Red Hat OpenShift Container Platform to perform the installation steps described.

- **cert manager configured**: 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. By default, Astra Control Center installs its own cert manager during installation.

- **Kubernetes ingress controller**: If you have a Kubernetes ingress controller that manages external access to services, such as load balancing in a cluster, you need to set it up for use with Astra Control Center:

  a. Create the operator namespace:

  ```
  oc create namespace netapp-acc-operator
  ```

  b. Complete setup for your ingress controller type.

Steps

- Download and extract Astra Control Center
- Install the NetApp Astra kubectl plugin
- Add the images to your local registry
- Find the operator install page
- Install the operator
- Install Astra Control Center

Download and extract Astra Control Center

1. Go to the Astra Control Center downloads page on the NetApp Support Site.

2. Download the bundle containing Astra Control Center (`astra-control-center-[version].tar.gz`).

3. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (`astra-control-center-certs-[version].tar.gz`) to verify the signature of the bundle:

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

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

The output will show `Verified OK` after successful verification.

4. Extract the images from the Astra Control Center bundle:

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

Install the NetApp Astra kubectl plugin

You can use the NetApp Astra kubectl command line plugin to push images to a local Docker repository.

Before you begin

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:

   (i) 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 root directory of the tarball. You should see this file and directory:

```
acc.manifest.bundle.yaml
acc/
```

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

   ◦ Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.bundle.yaml`).

   ◦ Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, "https://<docker-registry>".

   ◦ 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. Change to the root directory of the tarball. You should see this file and directory:

```
acc.manifest.bundle.yaml
acc/
```

b. Log in to your registry:

```
podman login <YOUR_REGISTRY>
```

c. Prepare and run one of the following scripts that is customized for the version of Podman you use. Substitute <MY_FULL_REGISTRY_PATH> with the URL of your repository that includes any sub-directories.

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

> ⓘ   The image path the script creates should resemble the following, depending on your
> registry configuration:

```
https://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/image:version
```

Find the operator install page

1. Complete one of the following procedures to access the operator install page:

◦ From Red Hat Openshift web console:

   a. Log in to the OpenShift Container Platform UI.

   b. From the side menu, select **Operators > OperatorHub**.

   c. Search for and select the NetApp Astra Control Center operator.
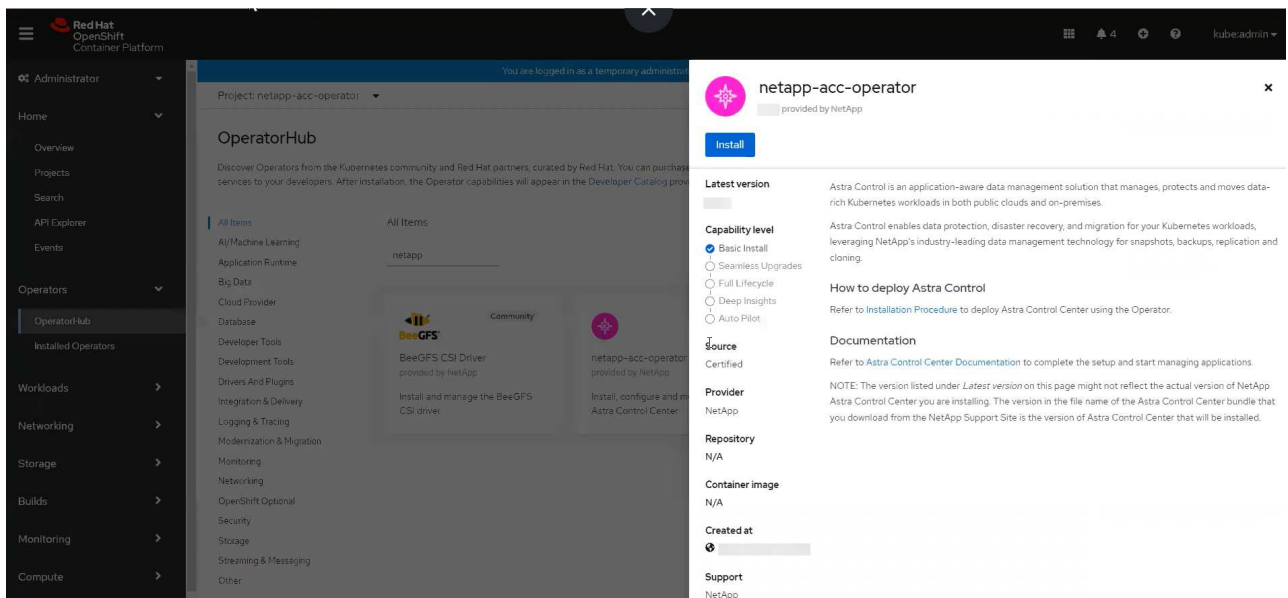
◦ From Red Hat Ecosystem Catalog:

   a. Select the NetApp Astra Control Center operator.

   b. Select **Deploy and Use**.
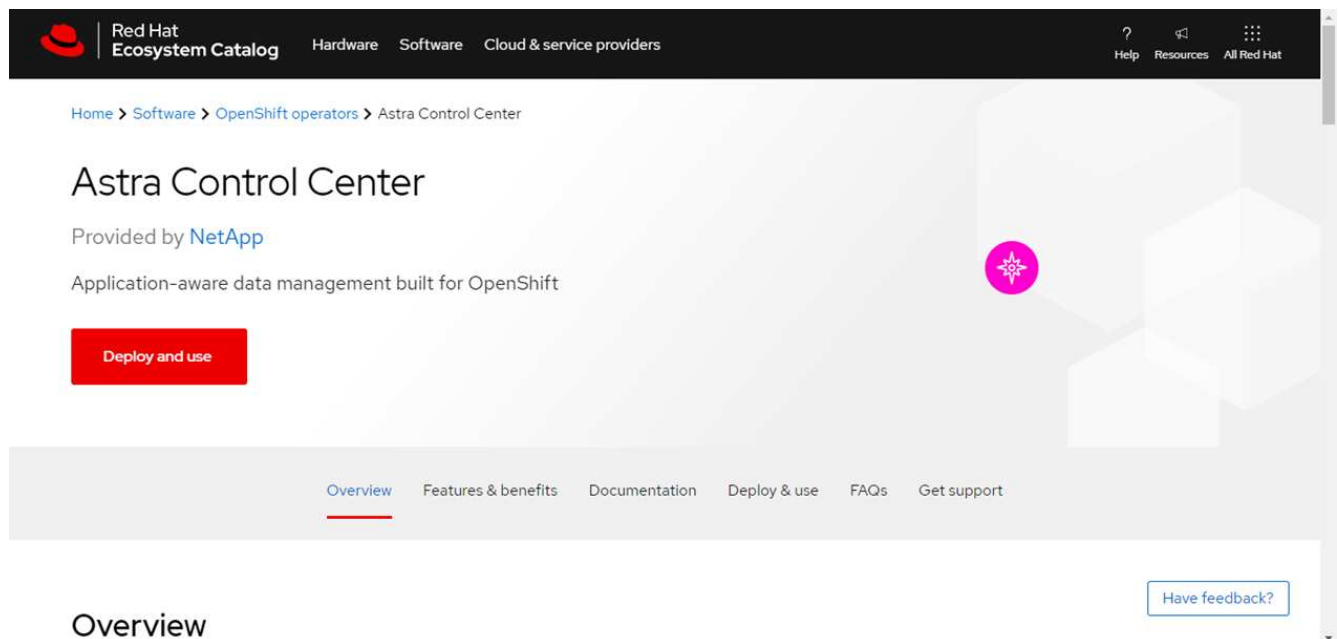
Install the operator

1. Complete the **Install Operator** page and install the operator:

ⓘ The operator will be available in all cluster namespaces.

a. Select the operator namespace or `netapp-acc-operator` namespace will be created automatically as part of the operator installation.

b. Select a manual or automatic approval strategy.

ⓘ Manual approval is recommended. You should only have a single operator instance running per cluster.

c. Select **Install**.

ⓘ If you selected a manual approval strategy, you will be prompted to approve the manual install plan for this operator.

2. From the console, go to the OperatorHub menu and confirm that the operator installed successfully.

Install Astra Control Center

1. From the console within the **Astra Control Center** tab of the Astra Control Center operator, select **Create AstraControlCenter**.

2. Complete the `Create AstraControlCenter` form field:

a. Keep or adjust the Astra Control Center name.

b. Add labels for the Astra Control Center.

c. Enable or disable Auto Support. Retaining Auto Support functionality is recommended.

d. Enter the Astra Control Center FQDN or IP address. Do not enter `http://` or `https://` in the address field.

e. Enter the Astra Control Center version; for example, 23.04.2-7.

f. Enter an account name, email address, and admin last name.

g. Choose a volume reclaim policy of `Retain`, `Recycle`, or `Delete`. The default value is `Retain`.

h. Select the scaleSize of the installation.

> **(i)** By default, Astra will use High Availability (HA) `scaleSize` of `Medium`, which deploys most services in HA and deploys multiple replicas for redundancy. With `scaleSize` as `Small`, Astra will reduce the number of replicas for all services except for essential services to reduce consumption.

i. Select the ingress type:

- **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.

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

j. In **Image Registry**, enter your local container image registry path. Do not enter `http://` or `https://` in the address field.

k. If you use an image registry that requires authentication, enter the image secret.

> **(i)** If you use a registry that requires authentication, create a secret on the cluster.

l. Enter the admin first name.

m. Configure resources scaling.

n. Provide the default storage class.

> **(i)** If a default storage class is configured, ensure that it is the only storage class that has the default annotation.

o. Define CRD handling preferences.

3. Select the YAML view to review the settings you have selected.

4. Select `Create`.

Create a registry secret

If you use a registry that requires authentication, create a secret on the Openshift cluster and enter the secret name in the `Create AstraControlCenter` form field.

1. Create a namespace for the Astra Control Center operator:

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. Create a secret in this namespace:

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

> (i)   Astra Control supports Docker registry secrets only.

3. Complete the remaining fields in the Create AstraControlCenter form field.

What's next

Complete the remaining steps to verify that Astra Control Center installed successfully, set up an ingress controller (optional), and log in to the UI. Additionally, you will need to perform setup tasks after completing installation.

Install Astra Control Center with a Cloud Volumes ONTAP storage backend

With Astra Control Center, you can manage your apps in a hybrid cloud environment with self-managed Kubernetes clusters and Cloud Volumes ONTAP instances. You can deploy Astra Control Center in your on-premise Kubernetes clusters or in one of the self-managed Kubernetes clusters in the cloud environment.

With one of these deployments, you can perform app data management operations using Cloud Volumes ONTAP as a storage backend. You can also configure an S3 bucket as the backup target.

To install Astra Control Center in Amazon Web Services (AWS), Google Cloud Platform (GCP) and Microsoft Azure with a Cloud Volumes ONTAP storage backend, perform the following steps depending on your cloud environment.

- Deploy Astra Control Center in Amazon Web Services
- Deploy Astra Control Center in Google Cloud Platform
- Deploy Astra Control Center in Microsoft Azure

You can manage your apps in distributions with self-managed Kubernetes clusters, such with OpenShift Container Platform (OCP). Only self-managed OCP clusters are validated for deploying Astra Control Center.

Deploy Astra Control Center in Amazon Web Services

You can deploy Astra Control Center on a self-managed Kubernetes cluster hosted on an Amazon Web Services (AWS) public cloud.

What you'll need for AWS

Before you deploy Astra Control Center in AWS, you will need the following items:

- Astra Control Center license. Refer to Astra Control Center licensing requirements.
- Meet Astra Control Center requirements.
- NetApp Cloud Central account
- If using OCP, Red Hat OpenShift Container Platform (OCP) permissions (on namespace level to create pods)
- AWS credentials, Access ID and Secret Key with permissions that enable you to create buckets and connectors
- AWS account Elastic Container Registry (ECR) access and login
- AWS hosted zone and Route 53 entry required to access the Astra Control UI

Operational environment requirements for AWS

Astra Control Center requires the following operational environment for AWS:

- Red Hat OpenShift Container Platform 4.8

> (i) Ensure that the operating environment you choose to host Astra Control Center meets the basic resource requirements outlined in the environment's official documentation.

Astra Control Center requires the following resources in addition to the environment's resource requirements:

Component	Requirement
Backend NetApp Cloud Volumes ONTAP storage capacity	At least 300GB available
Worker nodes (AWS EC2 requirement)	At least 3 worker nodes total, with 4 vCPU cores and 12GB RAM each
Load balancer	Service type "LoadBalancer" available for ingress traffic to be sent to services in the operational environment cluster
FQDN	A method for pointing the FQDN of Astra Control Center to the load balanced IP address
Astra Trident (installed as part of the Kubernetes cluster discovery in NetApp BlueXP, formerly Cloud Manager)	Astra Trident 21.04 or newer installed and configured and NetApp ONTAP version 9.5 or newer as a storage backend

Component	Requirement
Image registry	You must have an existing private registry, such as AWS Elastic Container Registry, to which you can push Astra Control Center build images. You need to provide the URL of the image registry where you will upload the images.
	ⓘ The Astra Control Center hosted cluster and the managed cluster must have access to the same image registry to be able to back up and restore apps using the Restic-based image.
Astra Trident / ONTAP configuration	Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP Kubernetes storage classes that are created when you import your Kubernetes cluster into NetApp BlueXP (formerly Cloud Manager). These are provided by Astra Trident:
	• `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io`
	• `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io`
	• `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io`
	• `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

ⓘ   These requirements assume that Astra Control Center is the only application running in the operational environment. If the environment is running additional applications, adjust these minimum requirements accordingly.

ⓘ   The AWS registry token expires in 12 hours, after which you will have to renew the Docker image registry secret.

Overview of deployment for AWS

Here is an overview of the process to install Astra Control Center for AWS with Cloud Volumes ONTAP as a storage backend.

Each of these steps is explained in more detail below.

1. Ensure that you have sufficient IAM permissions.
2. Install a RedHat OpenShift cluster on AWS.
3. Configure AWS.
4. Configure NetApp BlueXP for AWS.
5. Install Astra Control Center for AWS.

Ensure that you have sufficient IAM permissions

Ensure that you have sufficient IAM roles and permissions that enable you to install a RedHat OpenShift cluster and a NetApp BlueXP (formerly Cloud Manager) Connector.

See Initial AWS credentials.

Install a RedHat OpenShift cluster on AWS

Install a RedHat OpenShift Container Platform cluster on AWS.

For installation instructions, see Installing a cluster on AWS in OpenShift Container Platform.

Configure AWS

Next, configure AWS to create a virtual network, set up EC2 compute instances, create an AWS S3 bucket, create an Elastic Container Register (ECR) to host the Astra Control Center images, and push the images to this registry.

Follow the AWS documentation to complete the following steps. See AWS installation documentation.

1. Create an AWS virtual network.

2. Review the EC2 compute instances. This can be a bare metal server or VMs in AWS.

3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in AWS to meet the Astra requirements. Refer to Astra Control Center requirements.

4. Create at least one AWS S3 bucket to store your backups.

5. Create an AWS Elastic Container Registry (ECR) to host all the ACC images.

    > ⓘ If you do not create the ECR, Astra Control Center cannot access monitoring data from a cluster containing Cloud Volumes ONTAP with an AWS backend. The issue is caused when the cluster you try to discover and manage using Astra Control Center does not have AWS ECR access.

6. Push the ACC images to your defined registry.

    > ⓘ The AWS Elastic Container Registry (ECR) token expires after 12 hours and causes cross-cluster clone operations to fail. This issue occurs when managing a storage backend from Cloud Volumes ONTAP configured for AWS. To correct this issue, authenticate with the ECR again and generate a new secret for clone operations to resume successfully.

Here's an example of an AWS deployment:
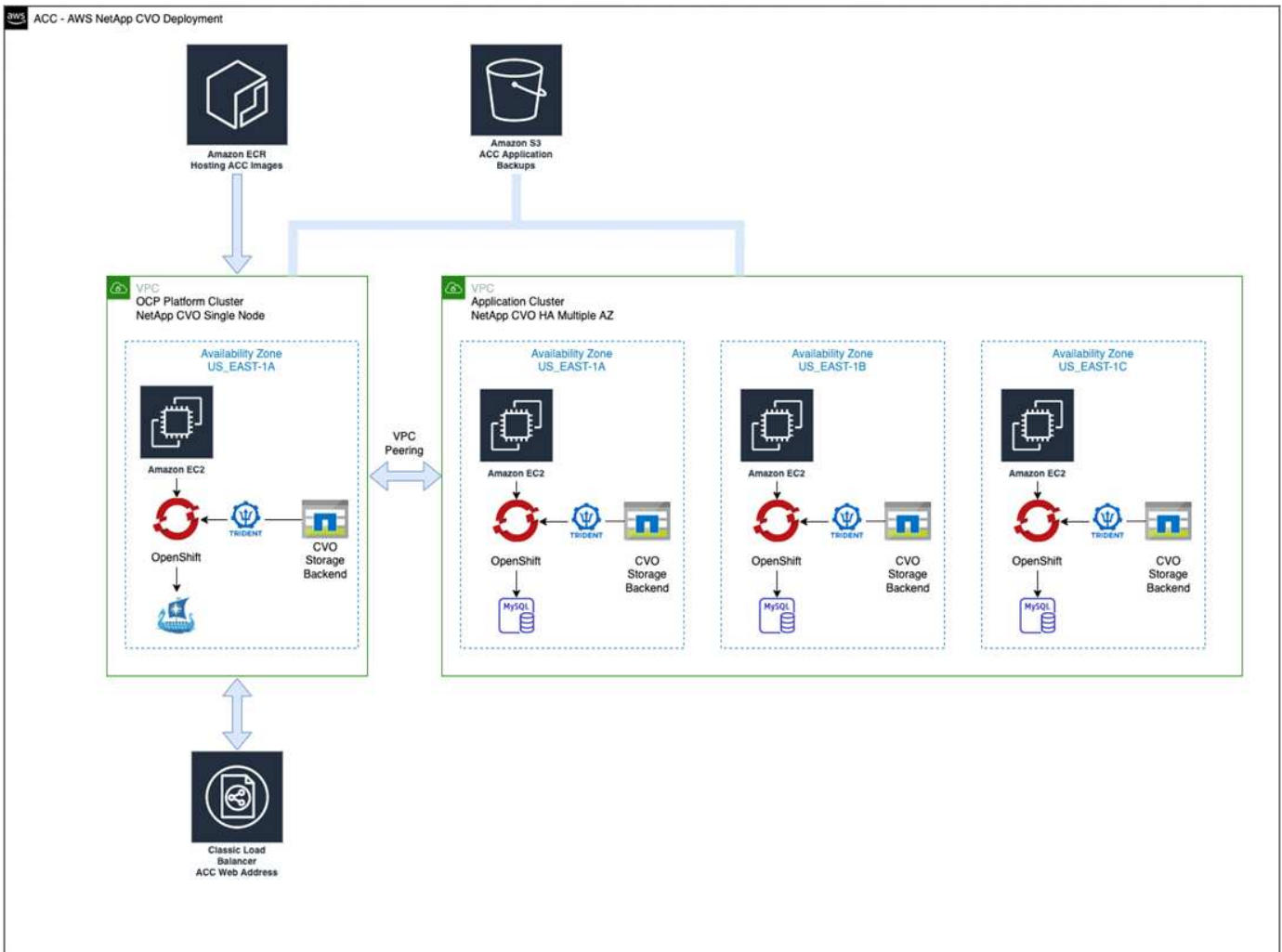
Configure NetApp BlueXP for AWS

Using NetApp BlueXP (formerly Cloud Manager), create a workspace, add a connector to AWS, create a working environment, and import the cluster.

Follow the BlueXP documentation to complete the following steps. See the following:

- Getting started with Cloud Volumes ONTAP in AWS.
- Create a connector in AWS using BlueXP

Steps

1. Add your credentials to BlueXP.

2. Create a workspace.

3. Add a connector for AWS. Choose AWS as the Provider.

4. Create a working environment for your cloud environment.

    a. Location: "Amazon Web Services (AWS)"

    b. Type: "Cloud Volumes ONTAP HA"

5. Import the OpenShift cluster. The cluster will connect to the working environment you just created.

    a. View the NetApp cluster details by selecting **K8s** > **Cluster list** > **Cluster Details**.

b. In the upper right corner, note the Astra Trident version.

c. Note the Cloud Volumes ONTAP cluster storage classes showing NetApp as the provisioner.

This imports your Red Hat OpenShift cluster and assigns it a default storage class. You select the storage class.
Astra Trident is automatically installed as part of the import and discovery process.

6. Note all the persistent volumes and volumes in this Cloud Volumes ONTAP deployment.

> Cloud Volumes ONTAP can operate as a single node or in High Availability. If HA is enabled, note the HA status and node deployment status running in AWS.

Install Astra Control Center for AWS

Follow the standard Astra Control Center installation instructions.

> AWS uses the Generic S3 bucket type.

Deploy Astra Control Center in Google Cloud Platform

You can deploy Astra Control Center on a self-managed Kubernetes cluster hosted on a Google Cloud Platform (GCP) public cloud.

What you'll need for GCP

Before you deploy Astra Control Center in GCP, you will need the following items:

- Astra Control Center license. Refer to Astra Control Center licensing requirements.
- Meet Astra Control Center requirements.
- NetApp Cloud Central account
- If using OCP, Red Hat OpenShift Container Platform (OCP) 4.10
- If using OCP, Red Hat OpenShift Container Platform (OCP) permissions (on namespace level to create pods)
- GCP Service Account with permissions that enable you to create buckets and connectors

Operational environment requirements for GCP

> Ensure that the operating environment you choose to host Astra Control Center meets the basic resource requirements outlined in the environment's official documentation.

Astra Control Center requires the following resources in addition to the environment's resource requirements:

Component	Requirement
Backend NetApp Cloud Volumes ONTAP storage capacity	At least 300GB available
Worker nodes (GCP compute requirement)	At least 3 worker nodes total, with 4 vCPU cores and 12GB RAM each

Component	Requirement
Load balancer	Service type "LoadBalancer" available for ingress traffic to be sent to services in the operational environment cluster
FQDN (GCP DNS zone)	A method for pointing the FQDN of Astra Control Center to the load balanced IP address
Astra Trident (installed as part of the Kubernetes cluster discovery in NetApp BlueXP, formerly Cloud Manager)	Astra Trident 21.04 or newer installed and configured and NetApp ONTAP version 9.5 or newer as a storage backend
Image registry	You must have an existing private registry, such as Google Container Registry, to which you can push Astra Control Center build images. You need to provide the URL of the image registry where you will upload the images. ⓘ You need to enable anonymous access to pull Restic images for backups.
Astra Trident / ONTAP configuration	Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP Kubernetes storage classes that are created when you import your Kubernetes cluster into NetApp BlueXP. These are provided by Astra Trident: • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

ⓘ These requirements assume that Astra Control Center is the only application running in the operational environment. If the environment is running additional applications, adjust these minimum requirements accordingly.

Overview of deployment for GCP

Here is an overview of the process to install Astra Control Center on a self-managed OCP cluster in GCP with Cloud Volumes ONTAP as a storage backend.

Each of these steps is explained in more detail below.

1. Install a RedHat OpenShift cluster on GCP.
2. Create a GCP Project and Virtual Private Cloud.

3. Ensure that you have sufficient IAM permissions.

4. Configure GCP.

5. Configure NetApp BlueXP for GCP.

6. Install Astra Control Center for GCP.

Install a RedHat OpenShift cluster on GCP

The first step is to install a RedHat OpenShift cluster on GCP.

For installation instructions, see the following:

- Installing an OpenShift cluster in GCP
- Creating a GCP Service Account

Create a GCP Project and Virtual Private Cloud

Create at least one GCP Project and Virtual Private Cloud (VPC).

> ℹ️ OpenShift might create its own resource groups. In addition to these, you should also define a GCP VPC. Refer to OpenShift documentation.

You might want to create a platform cluster resource group and a target app OpenShift cluster resource group.

Ensure that you have sufficient IAM permissions

Ensure that you have sufficient IAM roles and permissions that enable you to install a RedHat OpenShift cluster and a NetApp BlueXP (formerly Cloud Manager) Connector.

See Initial GCP credentials and permissions.

Configure GCP

Next, configure GCP to create a VPC, set up compute instances, create a Google Cloud Object Storage, create an Google Container Register to host the Astra Control Center images, and push the images to this registry.

Follow the GCP documentation to complete the following steps. See Installing OpenShift cluster in GCP.

1. Create a GCP Project and VPC in the GCP that you plan on using for the OCP cluster with CVO backend.

2. Review the compute instances. This can be a bare metal server or VMs in GCP.

3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in GCP to meet the Astra requirements. Refer to Astra Control Center requirements.

4. Create at least one GCP Cloud Storage Bucket to store your backups.

5. Create a secret, which is required for bucket access.

6. Create a Google Container Registry to host all the Astra Control Center images.

7. Set up Google Container Registry access for Docker push/pull for all the Astra Control Center images.

   Example: ACC images can be pushed to this registry by entering the following script:

```
gcloud auth activate-service-account <service account email address>
--key-file=<GCP Service Account JSON file>
```

This script requires an Astra Control Center manifest file and your Google Image Registry location.

Example:

```
manifestfile=astra-control-center-<version>.manifest
GCP_CR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

8. Set up DNS zones.

Configure NetApp BlueXP for GCP

Using NetApp BlueXP (formerly Cloud Manager), create a workspace, add a connector to GCP, create a working environment, and import the cluster.

Follow the BlueXP documentation to complete the following steps. See Getting started with Cloud Volumes ONTAP in GCP.

Before you begin

- Access to the GCP Service Account with the required IAM permissions and roles

Steps

1. Add your credentials to BlueXP. See Adding GCP accounts.

2. Add a connector for GCP.

   a. Choose "GCP" as the Provider.

   b. Enter GCP credentials. See Creating a connector in GCP from BlueXP.

   c. Ensure that the connector is running and switch to that connector.

3. Create a working environment for your cloud environment.

   a. Location: "GCP"

   b. Type: "Cloud Volumes ONTAP HA"

4. Import the OpenShift cluster. The cluster will connect to the working environment you just created.

   a. View the NetApp cluster details by selecting **K8s** > **Cluster list** > **Cluster Details**.

b. In the upper right corner, note the Trident version.

c. Note the Cloud Volumes ONTAP cluster storage classes showing "NetApp" as the provisioner.

This imports your Red Hat OpenShift cluster and assigns it a default storage class. You select the storage class.
Astra Trident is automatically installed as part of the import and discovery process.

5. Note all the persistent volumes and volumes in this Cloud Volumes ONTAP deployment.

> Cloud Volumes ONTAP can operate as a single node or in High Availability (HA). If HA is enabled, note the HA status and node deployment status running in GCP.

Install Astra Control Center for GCP

Follow the standard Astra Control Center installation instructions.

> GCP uses the Generic S3 bucket type.

1. Generate the Docker Secret to pull images for the Astra Control Center installation:

```
kubectl create secret docker-registry <secret name> --docker
-server=<Registry location> --docker-username=_json_key --docker
-password="$(cat <GCP Service Account JSON file>)" --namespace=pcloud
```

Deploy Astra Control Center in Microsoft Azure

You can deploy Astra Control Center on a self-managed Kubernetes cluster hosted on a Microsoft Azure public cloud.

What you'll need for Azure

Before you deploy Astra Control Center in Azure, you will need the following items:

- Astra Control Center license. Refer to Astra Control Center licensing requirements.
- Meet Astra Control Center requirements.
- NetApp Cloud Central account
- If using OCP, Red Hat OpenShift Container Platform (OCP) 4.8
- If using OCP, Red Hat OpenShift Container Platform (OCP) permissions (on namespace level to create pods)
- Azure credentials with permissions that enable you to create buckets and connectors

Operational environment requirements for Azure

Ensure that the operating environment you choose to host Astra Control Center meets the basic resource requirements outlined in the environment's official documentation.

Astra Control Center requires the following resources in addition to the environment's resource requirements:

Refer to Astra Control Center operational environment requirements.

Component	Requirement
Backend NetApp Cloud Volumes ONTAP storage capacity	At least 300GB available
Worker nodes (Azure compute requirement)	At least 3 worker nodes total, with 4 vCPU cores and 12GB RAM each
Load balancer	Service type "LoadBalancer" available for ingress traffic to be sent to services in the operational environment cluster
FQDN (Azure DNS zone)	A method for pointing the FQDN of Astra Control Center to the load balanced IP address
Astra Trident (installed as part of the Kubernetes cluster discovery in NetApp BlueXP)	Astra Trident 21.04 or newer installed and configured and NetApp ONTAP version 9.5 or newer will be used as a storage backend
Image registry	You must have an existing private registry, such as Azure Container Registry (ACR), to which you can push Astra Control Center build images. You need to provide the URL of the image registry where you will upload the images. ⓘ You need to enable anonymous access to pull Restic images for backups.
Astra Trident / ONTAP configuration	Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP Kubernetes storage classes that are created when you import your Kubernetes cluster into NetApp BlueXP. These are provided by Astra Trident: • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

ⓘ These requirements assume that Astra Control Center is the only application running in the operational environment. If the environment is running additional applications, adjust these minimum requirements accordingly.

Overview of deployment for Azure

Here is an overview of the process to install Astra Control Center for Azure.

Each of these steps is explained in more detail below.

1. Install a RedHat OpenShift cluster on Azure.

2. Create Azure resource groups.

3. Ensure that you have sufficient IAM permissions.

4. Configure Azure.

5. Configure NetApp BlueXP (formerly Cloud Manager) for Azure.

6. Install and configure Astra Control Center for Azure.

Install a RedHat OpenShift cluster on Azure

The first step is to install a RedHat OpenShift cluster on Azure.

For installation instructions, see the following:

- Installing OpenShift cluster on Azure.

- Installing an Azure account.

Create Azure resource groups

Create at least one Azure resource group.

> ⓘ OpenShift might create its own resource groups. In addition to these, you should also define
> Azure resource groups. Refer to OpenShift documentation.

You might want to create a platform cluster resource group and a target app OpenShift cluster resource group.

Ensure that you have sufficient IAM permissions

Ensure that you have sufficient IAM roles and permissions that enable you to install a RedHat OpenShift cluster and a NetApp BlueXP Connector.

See Azure credentials and permissions.

Configure Azure

Next, configure Azure to create a virtual network, set up compute instances, create an Azure Blob container, create an Azure Container Register (ACR) to host the Astra Control Center images, and push the images to this registry.

Follow the Azure documentation to complete the following steps. See Installing OpenShift cluster on Azure.

1. Create an Azure virtual network.

2. Review the compute instances. This can be a bare metal server or VMs in Azure.

3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in Azure to meet the Astra requirements. Refer to Astra Control Center requirements.

4. Create at least one Azure Blob container to store your backups.

5. Create a storage account. You will need a storage account to create a container to be used as a bucket in Astra Control Center.

6. Create a secret, which is required for bucket access.

7. Create an Azure Container Registry (ACR) to host all the Astra Control Center images.

8. Set up ACR access for Docker push/pull all the Astra Control Center images.

9. Push the ACC images to this registry by entering the following script:

```
az acr login -n <AZ ACR URL/Location>
This script requires ACC manifest file and your Azure ACR location.
```

Example:

```
manifestfile=astra-control-center-<version>.manifest
AZ_ACR_REGISTRY=<target image repository>
ASTRA_REGISTRY=<source ACC image repository>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRYY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

10. Set up DNS zones.

Configure NetApp BlueXP (formerly Cloud Manager) for Azure

Using BlueXP (formerly Cloud Manager), create a workspace, add a connector to Azure, create a working environment, and import the cluster.

Follow the BlueXP documentation to complete the following steps. See Getting started with BlueXP in Azure.

Before you begin
Access to the Azure account with the required IAM permissions and roles

Steps
1. Add your credentials to BlueXP.

2. Add a connector for Azure. See BlueXP policies.

   a. Choose **Azure** as the Provider.

   b. Enter Azure credentials, including the application ID, client secret, and directory (tenant) ID.

      See Creating a connector in Azure from BlueXPr.

3. Ensure that the connector is running and switch to that connector.
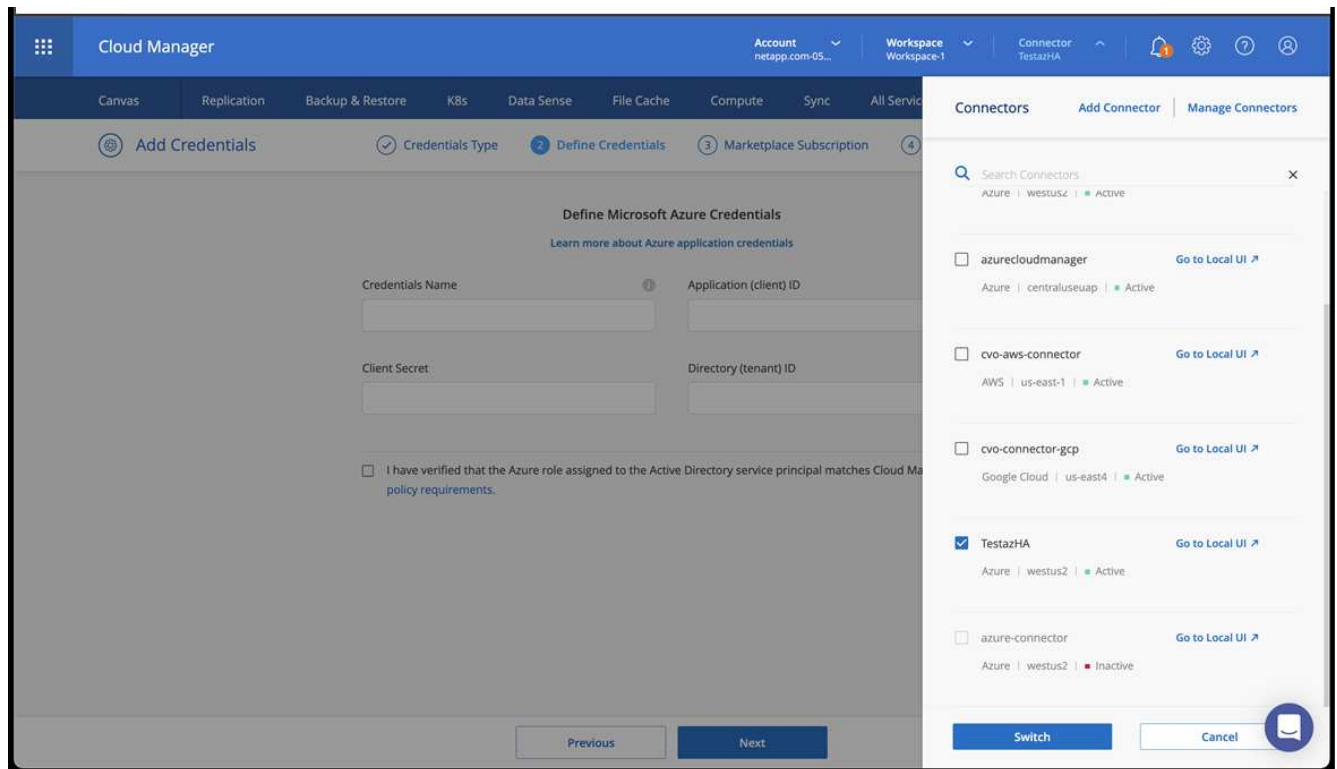
4. Create a working environment for your cloud environment.

    a. Location: "Microsoft Azure".

    b. Type: "Cloud Volumes ONTAP HA".
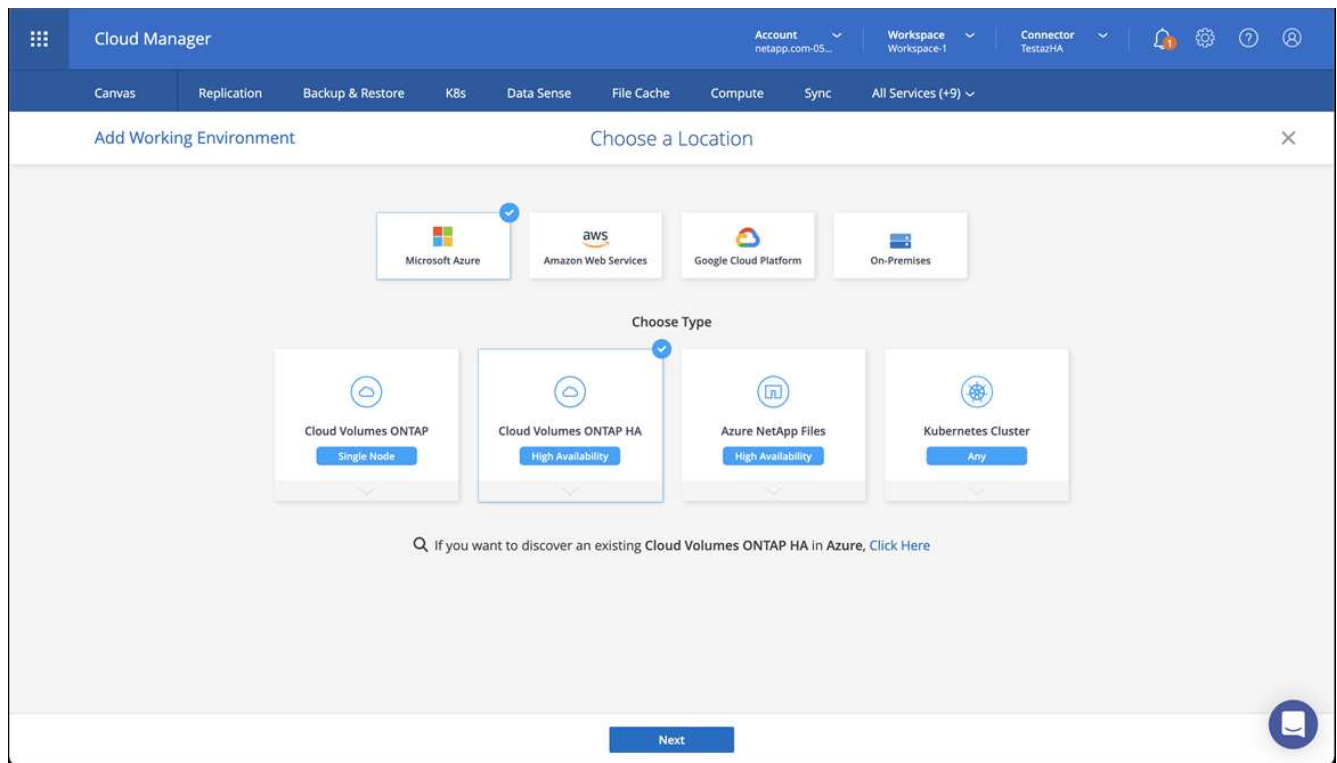
5. Import the OpenShift cluster. The cluster will connect to the working environment you just created.

    a. View the NetApp cluster details by selecting **K8s** > **Cluster list** > **Cluster Details**.
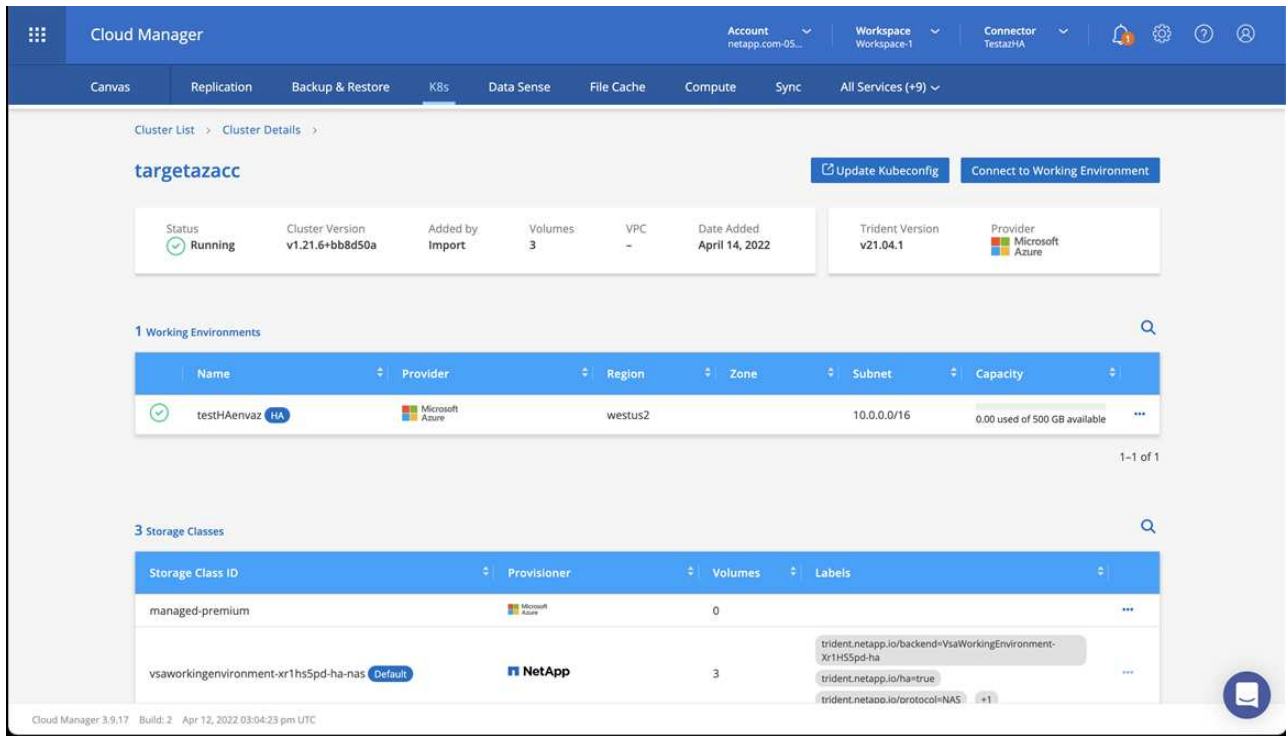
b. In the upper right corner, note the Astra Trident version.

c. Note the Cloud Volumes ONTAP cluster storage classes showing NetApp as the provisioner.

This imports your Red Hat OpenShift cluster and assigns a default storage class. You select the storage class.
Astra Trident is automatically installed as part of the import and discovery process.

6. Note all the persistent volumes and volumes in this Cloud Volumes ONTAP deployment.

7. Cloud Volumes ONTAP can operate as a single node or in High Availability. If HA is enabled, note the HA status and node deployment status running in Azure.

Install and configure Astra Control Center for Azure

Install Astra Control Center with the standard installation instructions.

Using Astra Control Center, add an Azure bucket. Refer to Set up Astra Control Center and add buckets.

Configure Astra Control Center after installation

Depending on your environment, there might be additional configuration needed after you install Astra Control Center.

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. If your environment is configured this way, you need to remove those resources 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` (or custom-named) namespace:

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

Response:

```
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 or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. Get the limit ranges in the `netapp-acc` (or custom-named) namespace:

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

Response:

```
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 or custom
namespace]
```

Enable network communication between namespaces

Some environments use NetworkPolicy constructs to restrict traffic between namespaces. The Astra Control Center operator and Astra Control Center are 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 or custom namespace]
```

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 or custom
namespace]
```

3. Apply the following resource file to configure the `acc-avp-network-policy` object to allow Astra plugin 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 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:

```
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
```

Add a custom TLS certificate

Astra Control Center uses a self-signed TLS certificate by default for ingress controller traffic (only in certain configurations) and web UI authentication with web browsers. You can remove the existing self-signed TLS certificate and replace it with a TLS certificate signed by a Certificate Authority (CA).

ⓘ The default, self-signed certificate is used for two types of connections:

- HTTPS connections to the Astra Control Center web UI

- Ingress controller traffic (only if the `ingressType: "AccTraefik"` property was set in the `astra_control_center.yaml` file during Astra Control Center installation)

Replacing the default TLS certificate replaces the certificate used for authentication for these connections.

Before you begin
- 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 using the command line

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:

```yaml
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.