Get started

Astra Control Center

NetApp
August 11, 2025

Table of Contents

Get started

Learn about Astra Control

Astra Control is a Kubernetes application data lifecycle management solution that simplifies operations for stateful applications. Easily protect, back up, replicate, and migrate Kubernetes workloads, and instantly create working application clones.

Features

Astra Control offers critical capabilities for Kubernetes application data lifecycle management:

- Automatically manage persistent storage
- Create application-aware, on-demand snapshots and backups
- Automate policy-driven snapshot and backup operations
- Migrate applications and data from one Kubernetes cluster to another
- Replicate applications to a remote system using NetApp SnapMirror technology (Astra Control Center)
- Clone applications from staging to production
- Visualize application health and protection status
- Work with a web UI or an API to implement your backup and migration workflows

Deployment models

Astra Control is available in two deployment models:

- **Astra Control Service**: A NetApp-managed service that provides application-aware data management of Kubernetes clusters in multiple cloud provider environments, as well as self-managed Kubernetes clusters.
- **Astra Control Center**: Self-managed software that provides application-aware data management of Kubernetes clusters running in your on-premises environment. Astra Control Center can also be installed on multiple cloud provider environments with a NetApp Cloud Volumes ONTAP storage backend.

	Astra Control Service	**Astra Control Center**
How is it offered?	As a fully managed cloud service from NetApp	As software that you can download, install, and manage
Where is it hosted?	On a public cloud of NetApp's choice	On your own Kubernetes cluster
How is it updated?	Managed by NetApp	You manage any updates

	Astra Control Service	Astra Control Center
What are the supported Kubernetes distributions?	• **Cloud providers** ◦ Amazon Web Services ▪ Amazon Elastic Kubernetes Service (EKS) ◦ Google Cloud ▪ Google Kubernetes Engine (GKE) ◦ Microsoft Azure ▪ Azure Kubernetes Service (AKS) • **Self-managed clusters** ◦ Kubernetes (Upstream) ◦ Rancher Kubernetes Engine (RKE) ◦ Red Hat OpenShift Container Platform • **On-premises clusters** ◦ Red Hat OpenShift Container Platform on-premises	• Azure Kubernetes Service on Azure Stack HCI • Google Anthos • Kubernetes (Upstream) • Rancher Kubernetes Engine (RKE) • Red Hat OpenShift Container Platform

	Astra Control Service	Astra Control Center
What are the supported storage backends?	• **Cloud providers** ◦ Amazon Web Services ▪ Amazon EBS ▪ Amazon FSx for NetApp ONTAP ▪ Cloud Volumes ONTAP ◦ Google Cloud ▪ Google Persistent Disk ▪ NetApp Cloud Volumes Service ▪ Cloud Volumes ONTAP ◦ Microsoft Azure ▪ Azure Managed Disks ▪ Azure NetApp Files ▪ Cloud Volumes ONTAP • **Self-managed clusters** ◦ Amazon EBS ◦ Azure Managed Disks ◦ Google Persistent Disk ◦ Cloud Volumes ONTAP ◦ NetApp MetroCluster ◦ Longhorn • **On-premises clusters** ◦ NetApp MetroCluster ◦ NetApp ONTAP AFF and FAS systems ◦ NetApp ONTAP Select ◦ Cloud Volumes ONTAP ◦ Longhorn	• NetApp ONTAP AFF and FAS systems • NetApp ONTAP Select • Cloud Volumes ONTAP • Longhorn

How Astra Control Service works

Astra Control Service is a NetApp-managed cloud service that is always on and updated with the latest capabilities. It utilizes several components to enable application data lifecycle management.

At a high level, Astra Control Service works like this:

- You get started with Astra Control Service by setting up your cloud provider and by registering for an Astra account.

    - For GKE clusters, Astra Control Service uses NetApp Cloud Volumes Service for Google Cloud or Google Persistent Disks as the storage backend for your persistent volumes.
    - For AKS clusters, Astra Control Service uses Azure NetApp Files or Azure managed disks as the storage backend for your persistent volumes.
    - For Amazon EKS clusters, Astra Control Service uses Amazon Elastic Block Store or Amazon FSx for NetApp ONTAP as the storage backend for your persistent volumes.
- You add your first Kubernetes compute to Astra Control Service. Astra Control Service then does the following:
    - Creates an object store in your cloud provider account, which is where backup copies are stored.

      In Azure, Astra Control Service also creates a resource group, a storage account, and keys for the Blob container.

    - Creates a new admin role and Kubernetes service account on the cluster.
    - Uses that new admin role to install link../concepts/architecture#astra-control-components[Astra Control Provisioner^] on the cluster and to create one or more storage classes.
    - If you use a NetApp cloud service storage offering as your storage backend, Astra Control Service uses Astra Control Provisioner to provision persistent volumes for your apps. If you use Amazon EBS or Azure managed disks as your storage backend, you need to install a provider-specific CSI driver. Installation instructions are provided in Set up Amazon Web Services and Set up Microsoft Azure with Azure managed disks.
- At this point, you can add apps to your cluster. Persistent volumes will be provisioned on the new default storage class.
- You then use Astra Control Service to manage these apps, and start creating snapshots, backups, and clones.

Astra Control's Free Plan enables you to manage up to 10 namespaces in your account. If you want to manage more than 10, then you'll need to set up billing by upgrading from the Free Plan to the Premium Plan.

How Astra Control Center works

Astra Control Center runs locally in your own private cloud.

Astra Control Center supports Kubernetes clusters with a Astra Control Provisioner-configured storage class with an ONTAP storage backend.

Limited (7-days of metrics) monitoring and telemetry is available in Astra Control Center and also exported to Kubernetes native monitoring tools (such as Prometheus and Grafana) through open metrics end points.

Astra Control Center is fully integrated into the AutoSupport and Active IQ Digital Advisor (also known as Digital Advisor) ecosystem to provide users and NetApp Support with troubleshooting and usage information.

You can try Astra Control Center out using a 90-day embedded evaluation license. While you are evaluating Astra Control Center, you can get support through email and community options. Additionally, you have access to Knowledgebase articles and documentation from the in-product support dashboard.

To install and use Astra Control Center, you'll need to meet certain requirements.

At a high level, Astra Control Center works like this:

- You install Astra Control Center in your local environment. Learn more about how to install Astra Control

Center.

- You complete some setup tasks such as these:
  - Set up licensing.
  - Add your first cluster.
  - Add storage backend that is discovered when you added the cluster.
  - Add an object store bucket that will store your app backups.

Learn more about how to set up Astra Control Center.

You can add apps to your cluster. Or, if you have some apps already in the cluster being managed, you can use Astra Control Center to manage them. Then, use Astra Control Center to create snapshots, backups, clones and replication relationships.

For more information

- Astra Control Service documentation
- Astra Control Center documentation
- Astra Trident documentation
- Astra Control API documentation
- ONTAP documentation

Astra Control Center requirements

Get started by verifying the readiness of your operational environment, application clusters, applications, licenses, and web browser. Ensure that your environment meets these requirements to deploy and operate Astra Control Center.

Supported host cluster Kubernetes environments

Astra Control Center has been validated with the following Kubernetes host environments:

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

Kubernetes distribution on host cluster	Supported versions
Azure Kubernetes Service on Azure Stack HCI	Azure Stack HCI 21H2 and 22H2 with AKS 1.24.11 through 1.26.6
Google Anthos	1.15 through 1.16 (See Google Anthos ingress requirements)
Kubernetes (Upstream)	1.27 to 1.29

Kubernetes distribution on host cluster	Supported versions
Rancher Kubernetes Engine (RKE)	RKE 1: Versions 1.24.17, 1.25.13, 1.26.8 with Rancher Manager 2.7.9 RKE 2: Versions 1.23.16 and 1.24.13 with Rancher Manager 2.6.13 RKE 2: Versions 1.24.17, 1.25.14, 1.26.9 with Rancher Manager 2.7.9
Red Hat OpenShift Container Platform	4.12 through 4.14

Host cluster resource requirements

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

> 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.

- **CPU extensions**: The CPUs in all nodes of the hosting environment must have AVX extensions enabled.
- **Worker nodes**: At least 3 worker nodes total, with 4 CPU cores and 12GB RAM each
- **VMware Tanzu Kubernetes Grid cluster requirements**: When hosting Astra Control Center on a VMware Tanzu Kubernetes Grid (TKG) or Tanzu Kubernetes Grid Integrated Edition (TKGi) cluster, keep in mind the following considerations.
  - The default VMware TKG and TKGi configuration file token expires ten hours after deployment. If you use Tanzu portfolio products, you must generate a Tanzu Kubernetes Cluster configuration file with a non-expiring token to prevent connection issues between Astra Control Center and managed application clusters. For instructions, visit the VMware NSX-T Data Center Product Documentation.
  - Use the `kubectl get nsxlbmonitors -A` command to see if you already have a service monitor configured to accept ingress traffic. If one exists, you should not install MetalLB, because the existing service monitor will override any new load balancer configuration.
  - Disable the TKG or TKGi default storage class enforcement on any application clusters intended to be managed by Astra Control. You can do this by editing the `TanzuKubernetesCluster` resource on the namespace cluster.
  - Be aware of specific requirements for Astra Control Provisioner when you deploy Astra Control Center in a TKG or TKGi environment:
    - The cluster must support privileged workloads.
    - The `--kubelet-dir` flag should be set to the location of kubelet directory. By default, this is `/var/vcap/data/kubelet`.
    - Specifying the kubelet location using `--kubelet-dir` is known to work for Trident Operator, Helm, and `tridentctl` deployments.

Service mesh requirements

It's strongly recommended that you install a supported vanilla version of the Istio service mesh on the Astra Control Center host cluster. Refer to supported releases for supported versions of Istio. Branded releases of Istio service mesh, such as OpenShift Service Mesh, are not validated with Astra Control Center.

To integrate Astra Control Center with the Istio service mesh installed on the host cluster, you must do the

integration as part of an Astra Control Center installation and not independent of this process.

> Installing and using Astra Control Center without configuring a service mesh on the host cluster has potentially serious security implications.

Astra Trident

If you intend to use Astra Trident instead of Astra Control Provisioner with this release, Astra Trident 23.04 and later versions are supported. Astra Control Center will require Astra Control Provisioner in future releases.

Astra Control Provisioner

To use Astra Control Provisioner advanced storage functionality, you must install Astra Trident 23.10 or later and enable Astra Control Provisioner functionality. To use the latest Astra Control Provisioner functionality, you'll need the most recent versions of Astra Trident and Astra Control Center.

- **Minimum Astra Control Provisioner version for use with Astra Control Center**: Astra Control Provisioner 23.10 or later installed and configured.

ONTAP configuration with Astra Trident

- **Storage class**: Configure at least one storage class on the cluster. If a default storage class is configured, ensure that it is the only storage class with the default designation.
- **Storage drivers and worker nodes**: Ensure that you configure the worker nodes in your cluster with the appropriate storage drivers so that the pods can interact with the backend storage. Astra Control Center supports the following ONTAP drivers provided by Astra Trident:
  - `ontap-nas`
  - `ontap-san`
  - `ontap-san-economy` (application replication is not available with this storage class type)
  - `ontap-nas-economy` (snapshots and application replication policies are not available with this storage class type)

Storage backends

Ensure that you have a supported backend with sufficient capacity.

- **Required storage backend capacity**: At least 500GB available
- **Supported backends**: Astra Control Center supports the following storage backends:
  - NetApp ONTAP 9.9.1 or later AFF, FAS, and ASA systems
  - NetApp ONTAP Select 9.9.1 or later
  - NetApp Cloud Volumes ONTAP 9.9.1 or later
  - (For Astra Control Center tech preview) NetApp ONTAP 9.10.1 or later for data protection operations provided as tech preview
  - Longhorn 1.5.0 or later
    - Requires the manual creation of a VolumeSnapshotClass object. Refer to the Longhorn documentation for instructions.
  - NetApp MetroCluster

        - Managed Kubernetes clusters must be in a stretch configuration.
    - Storage backends available with supported cloud providers

ONTAP licenses

To use Astra Control Center, verify that you have the following ONTAP licenses, depending on what you need to accomplish:

- FlexClone
- SnapMirror: Optional. Needed only for replication to remote systems using SnapMirror technology. Refer to SnapMirror license information.
- S3 license: Optional. Needed only for ONTAP S3 buckets

To check whether your ONTAP system has the required licenses, refer to Manage ONTAP licenses.

NetApp MetroCluster

When you use NetApp MetroCluster as a storage backend, you need to do the following:

- Specify an SVM management LIF as a backend option in the Astra Trident driver that you use
- Ensure that you have the appropriate ONTAP license

To configure the MetroCluster LIF, refer to these options and examples for each driver:

- SAN
- NAS

Astra Control Center license

Astra Control Center requires an Astra Control Center license. When you install Astra Control Center, an embedded 90-day evaluation license for 4,800 CPU units is already activated. If you need more capacity or different evaluation terms, or want to upgrade to a full license, you can obtain a different evaluation license or full license from NetApp. You need a license to protect your applications and data.

You can try Astra Control Center by signing up for a free trial. You can sign up by registering here.

To set up the license, refer to use a 90-day evaluation license.

To learn more about how licenses work, refer to Licensing.

Networking requirements

Configure your operational environment to ensure Astra Control Center can communicate properly. The following networking configurations are required:

- **FQDN address**: You must have an FQDN address for Astra Control Center.
- **Access to the internet**: You should determine whether you have outside access to the internet. If you do not, some functionality might be limited, such as sending support bundles to the NetApp Support Site.
- **Port access**: The operational environment that hosts Astra Control Center communicates using the following TCP ports. You should ensure that these ports are allowed through any firewalls, and configure firewalls to allow any HTTPS egress traffic originating from the Astra network. Some ports require

connectivity both ways between the environment hosting Astra Control Center and each managed cluster (noted where applicable).

> You can deploy Astra Control Center in a dual-stack Kubernetes cluster, and Astra Control Center can manage applications and storage backends that have been configured for dual-stack operation. For more information about dual-stack cluster requirements, see the Kubernetes documentation.

Source	Destination	Port	Protocol	Purpose
Client PC	Astra Control Center	443	HTTPS	UI / API access - Ensure this port is open in both directions between Astra Control Center and the system used to access Astra Control Center
Metrics consumer	Astra Control Center worker node	9090	HTTPS	Metrics data communication - ensure each managed cluster can access this port on the cluster hosting Astra Control Center (two-way communication required)
Astra Control Center	Amazon S3 storage bucket provider	443	HTTPS	Amazon S3 storage communication
Astra Control Center	NetApp AutoSupport (https://support.netapp.com)	443	HTTPS	NetApp AutoSupport communication
Astra Control Center	Managed Kubernetes cluster	443/6443 **NOTE**: The port that the managed cluster uses might vary depending on the cluster. Refer to the documentation from your cluster software vendor.	HTTPS	Communication with managed cluster - ensure this port is open both ways between the cluster hosting Astra Control Center and each managed cluster

Ingress for on-premises Kubernetes clusters

You can choose the type of network ingress Astra Control Center uses. By default, Astra Control Center deploys the Astra Control Center gateway (service/traefik) as a cluster-wide resource. Astra Control Center also supports using a service load balancer, if they are permitted in your environment. If you would rather use a service load balancer and you don't already have one configured, you can use the MetalLB load balancer to automatically 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.

The load balancer should use an IP address located in the same subnet as the Astra Control Center worker node IP addresses.

For more information, refer to Set up ingress for load balancing.

Google Anthos ingress requirements

When hosting Astra Control Center on a Google Anthos cluster, note that Google Anthos includes the MetalLB load balancer and the Istio ingress service by default, enabling you to simply use the generic ingress capabilities of Astra Control Center during installation. Refer to Astra Control Center installation documentation for details.

Supported web browsers

Astra Control Center supports recent versions of Firefox, Safari, and Chrome with a minimum resolution of 1280 x 720.

Additional requirements for application clusters

Keep in mind these requirements if you plan to use these Astra Control Center features:

- **Application cluster requirements**: Cluster management requirements
  - **Managed application requirements**: Application management requirements
  - **Additional requirements for application replication**: Replication prerequisites

What's next

View the quick start overview.

Quick start for Astra Control Center

Here's an overview of the steps needed to get started with Astra Control Center. The links within each step take you to a page that provides more details.

1 Review Kubernetes cluster requirements

Ensure that your environment meets these requirements:

Kubernetes cluster

- Ensure your host cluster meets operational environment requirements
- Configure ingress for load balancing on-premises Kubernetes clusters

Storage integration

- Ensure your environment includes Astra Control Provisioner
- Enable Astra Control Provisioner advanced management and storage provisioning features
- Prepare cluster worker nodes

- Configure storage backends
- Configure storage classes
- Install a volume snapshot controller
- Create a volume snapshot class

ONTAP credentials

- Configure ONTAP credentials

Download and install Astra Control Center

Complete these installation tasks:

- Download Astra Control Center from the NetApp Support Site downloads page
- Obtain the NetApp license file:
  - If you are evaluating Astra Control Center, an embedded evaluation license is already included
  - If you already purchased Astra Control Center, generate your license file
- Install Astra Control Center
- Perform additional optional configuration steps

Complete some initial setup tasks

Complete some basic tasks to get started:

- Add a license
- Prepare your environment for cluster management
- Add a cluster
- Add a storage backend
- Add a bucket

Use Astra Control Center

After you finish setting up Astra Control Center, use the Astra Control UI or the Astra Control API to begin managing and protecting apps:

- Manage accounts: Users, roles, LDAP, credentials, and more.
- Manage notifications
- Manage apps: Define resources to manage.
- Protect apps: Configure protection policies and replicate, clone, and migrate apps.

For more information

- Use the Astra Control API
- Upgrade Astra Control Center

- Get help with Astra Control

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 images and perform the following steps. You can use this procedure to install Astra Control Center in internet-connected or air-gapped environments.

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

Before you begin

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

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

- **Ensure healthy services**: Check that all API services are in a healthy state and available:

```
kubectl get apiservices
```

- **Ensure a routable FQDN**: The Astra FQDN you plan to use can be routed to the 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.
- **Configure cert manager**: 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.
- **(ONTAP SAN driver only) Enable multipath**: If you are using an ONTAP SAN driver, be sure that multipath is enabled on all your Kubernetes clusters.

You should also consider the following:

- **Get access to the NetApp Astra Control image registry**:

  You have the option to obtain installation images and functionality enhancements for Astra Control, such as

Astra Control Provisioner, from the NetApp image registry.

1. Record your Astra Control account ID that you'll need to log in to the registry.

   You can see your account ID in the Astra Control Service web UI. Select the figure icon at the top right of the page, select **API access**, and write down your account ID.

2. From the same page, select **Generate API token** and copy the API token string to the clipboard and save it in your editor.
3. Log into the Astra Control registry:

   ```
   docker login cr.astra.netapp.io -u <account-id> -p <api-token>
   ```

- **Install a service mesh for secure communications**: It is strongly recommended that Astra Control host cluster communications channels be secured using a supported service mesh.

  Integrating Astra Control Center with a service mesh can only be done during Astra Control Center installation and not independent of this process. Changing back from a meshed to an unmeshed environment is not supported.

  For Istio service mesh use, you'll need to do the following:

  - Add an `istio-injection:enabled` label to the Astra namespace prior to deploying Astra Control Center.
  - Use the `Generic` ingress setting and provide an alternative ingress for external load balancing.
  - For Red Hat OpenShift clusters, you need to define `NetworkAttachmentDefinition` on all associated Astra Control Center namespaces (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` for application clusters, or any custom namespaces that have been substituted).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

Steps

To install Astra Control Center, do the following steps:

- Download and extract Astra Control Center
- Complete additional steps if you use a 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

> 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.

Download and extract Astra Control Center

Download the Astra Control Center images from one of the following locations:

- **Astra Control Service image registry**: Use this option if you don't use a local registry with the Astra Control Center images or if you prefer this method to the bundle download from the NetApp Support Site.

- **NetApp Support Site**: Use this option if you use a local registry with the Astra Control Center images.

Astra Control image registry

1. Log in to Astra Control Service.
2. On the Dashboard, select **Deploy a self-managed instance of Astra Control**.
3. Follow the instructions to log in to the Astra Control image registry, pull the Astra Control Center installation image, and extract the image.

NetApp Support Site

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

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

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

Complete additional steps if you use a local registry

If you are planning to push the Astra Control Center bundle to your local registry, you need to use the NetApp Astra kubectl command line plugin.

Install the NetApp Astra kubectl plugin

Complete these steps to install the most recent NetApp Astra kubectl command line plugin.

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:

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

   ```
   ls kubectl-astra/
   ```

2. Move the file you need for your operating system and CPU architecture 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 registry

1. If you are planning to push the Astra Control Center bundle to your local registry, complete the appropriate step sequence for your container engine:

Docker

a. Change to the root directory of the tarball. You should see the `acc.manifest.bundle.yaml` file and these directories:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

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/
kubectl-astra/
acc.manifest.bundle.yaml
```

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=24.02.0-69
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=24.02.0-69
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://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. Change the directory:

```
cd manifests
```

Set up namespace and secret for registries with auth requirements

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

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

Before you complete the installation, be sure your kubeconfig is pointing to the cluster where you want to install Astra Control Center.

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
```

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

The placeholder `your_registry_path` should match the location of the images that you uploaded earlier (for example, `[Registry_URL]/netapp/astra/astracc/24.02.0-69`).

```
kubectl create secret docker-registry astra-registry-cred -n netapp-
acc-operator --docker-server=cr.astra.netapp.io --docker
-username=[astra_account_id] --docker-password=[astra_api_token]
```

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

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]
```

   d. Create a secret for the `netapp-acc` (or custom-named) namespace. Add Docker information and run one of the the appropriate command depending on your registry preference:

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

```
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]
```

Install the Astra Control Center operator

1. (Local registries only) If you are using a local registry, complete these steps:

   a. Open the Astra Control Center operator deployment YAML:

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   An annotated sample YAML follows these steps.

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

   ```
   imagePullSecrets: [{name: astra-registry-cred}]
   ```

   c. Change `ASTRA_IMAGE_REGISTRY` for the `kube-rbac-proxy` image to the registry path where you pushed the images in a previous step.

   d. Change `ASTRA_IMAGE_REGISTRY` for the `acc-operator-controller-manager` image to the registry path where you pushed the images in a previous step.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  strategy:
    type: Recreate
```

```
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: ASTRA_IMAGE_REGISTRY/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: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        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
```

2. Install the Astra Control Center operator:

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

Expand for sample response:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.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
```

3. 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	`24.02.0-69`

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'll participate in NetApp's pro-active support application, Digital Advisor, 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 Astra Control image registry that hosts all images required to deploy Astra Control Center. The value will be pre-populated, and no action is required unless you configured a local registry. For a local registry, replace this existing value with 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	`cr.astra.netapp.io` (default) `example.registry.com/astra` (local registry example)

Setting	Use	Guidance	Type	Example
`imageRegistry.secret`	Optional	The name of the Kubernetes secret used to authenticate with the image registry. The value will be pre-populated, and no action is required unless you configured a local registry and the string you entered for that registry in `imageRegistry.name` requires a secret. IMPORTANT: If you are using a local registry that does not require authorization, you must delete this `secret` line within `imageRegistry` or the installation will fail.	string	`astra-registry-cred`

storageClass

Setting	Guidance	Type	Example
`storageClass`	Change the `storageClass` value from `ontap-gold` to another storageClass resource as required by your installation. Run the command `kubectl get sc` to determine your existing configured storage classes. One of the Astra Control Provisioner-configured 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'll need to configure the ingress controller to expose Astra Control Center with a URL. IMPORTANT: If you intend to use a service mesh with Astra Control Center, you must select `Generic` as ingress type and set up your own ingress controller. **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	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

Add the following additional values to the Astra Control Center CR to prevent a known issue in installation:

```
additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

crds

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

Setting	Guidance	Type	Example
`crds.externalCertManager`	If you use an external cert manager, change `externalCertManager` 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.externalTraefik`	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)

Be sure that you have selected the correct storage class and ingress type for your configuration before completing installation.

sample astra_control_center.yaml

```
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: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  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]
   ```

2. If you are using a service mesh with Astra Control Center, add the following label to the `netapp-acc` or custom namespace:

   Your ingress type (`ingressType`) must be set to `Generic` in the Astra Control Center CR before proceeding with this command.

```
kubectl label ns [netapp-acc or custom namespace] istio-
injection:enabled
```

3. (Recommended) Enable strict MTLs for Istio service mesh:

```
kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
spec:
  mtls:
    mode: STRICT
EOF
```

4. 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]
```

> 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.

Expand for sample response

```
acc-helm-repo-5bd77c9ddd-8wxm2              1/1     Running     0
1h
activity-5bb474dc67-8l9ss                   1/1     Running     0
1h
activity-5bb474dc67-qbrtq                   1/1     Running     0
1h
api-token-authentication-6wbj2              1/1     Running     0
1h
api-token-authentication-9pgw6              1/1     Running     0
1h
api-token-authentication-tqf6d              1/1     Running     0
1h
asup-5495f44dbd-z4kft                       1/1     Running     0
1h
authentication-6fdd899858-5x45s             1/1     Running     0
1h
bucketservice-84d47487d-n9xgp               1/1     Running     0
1h
bucketservice-84d47487d-t5jhm               1/1     Running     0
1h
cert-manager-5dcb7648c4-hbldc               1/1     Running     0
1h
cert-manager-5dcb7648c4-nr9qf               1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-bk2tf    1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-pfnck    1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-ngz2x        1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-rwtbn        1/1     Running     0
1h
certificates-5f5b7b4dd-52tnj                1/1     Running     0
1h
certificates-5f5b7b4dd-gtjbx                1/1     Running     0
1h
certificates-expiry-check-28477260-dz5vw    0/1     Completed   0
1h
cloud-extension-6f58cc579c-lzfmv            1/1     Running     0
1h
cloud-extension-6f58cc579c-zw2km            1/1     Running     0
1h
cluster-orchestrator-79dd5c8d95-qjg92       1/1     Running     0
1h
```

```
composite-compute-85dc84579c-nz82f          1/1     Running     0
1h
composite-compute-85dc84579c-wx2z2          1/1     Running     0
1h
composite-volume-bff6f4f76-789nj            1/1     Running     0
1h
composite-volume-bff6f4f76-kwnd4            1/1     Running     0
1h
credentials-79fd64f788-m7m8f                1/1     Running     0
1h
credentials-79fd64f788-qnc6c                1/1     Running     0
1h
entitlement-f69cdbd77-4p2kn                 1/1     Running     0
1h
entitlement-f69cdbd77-hswm6                 1/1     Running     0
1h
features-7b9585444c-7xd7m                   1/1     Running     0
1h
features-7b9585444c-dcqwc                   1/1     Running     0
1h
fluent-bit-ds-crq8m                         1/1     Running     0
1h
fluent-bit-ds-gmgq8                         1/1     Running     0
1h
fluent-bit-ds-gzr4f                         1/1     Running     0
1h
fluent-bit-ds-j6sf6                         1/1     Running     0
1h
fluent-bit-ds-v4t9f                         1/1     Running     0
1h
fluent-bit-ds-x7j59                         1/1     Running     0
1h
graphql-server-6cc684fb46-2x8lr             1/1     Running     0
1h
graphql-server-6cc684fb46-bshbd             1/1     Running     0
1h
hybridauth-84599f79fd-fjc7k                 1/1     Running     0
1h
hybridauth-84599f79fd-s9pmn                 1/1     Running     0
1h
identity-95df98cb5-dvlmz                    1/1     Running     0
1h
identity-95df98cb5-krf59                    1/1     Running     0
1h
influxdb2-0                                 1/1     Running     0
1h
```

```
keycloak-operator-6d4d688697-cfq8b          1/1     Running     0
1h
krakend-5d5c8f4668-7bq8g                    1/1     Running     0
1h
krakend-5d5c8f4668-t8hbn                    1/1     Running     0
1h
license-689cdd4595-2gsc8                    1/1     Running     0
1h
license-689cdd4595-g6vwk                    1/1     Running     0
1h
login-ui-57bb599956-4fwgz                   1/1     Running     0
1h
login-ui-57bb599956-rhztb                   1/1     Running     0
1h
loki-0                                      1/1     Running     0
1h
metrics-facade-846999bdd4-f7jdm             1/1     Running     0
1h
metrics-facade-846999bdd4-lnsxl             1/1     Running     0
1h
monitoring-operator-6c9d6c4b8c-ggkrl        2/2     Running     0
1h
nats-0                                      1/1     Running     0
1h
nats-1                                      1/1     Running     0
1h
nats-2                                      1/1     Running     0
1h
natssync-server-6df7d6cc68-9v2gd            1/1     Running     0
1h
nautilus-64b7fbdd98-bsgwb                   1/1     Running     0
1h
nautilus-64b7fbdd98-djlhw                   1/1     Running     0
1h
openapi-864584bccc-75nlv                    1/1     Running     0
1h
openapi-864584bccc-zh6bx                    1/1     Running     0
1h
polaris-consul-consul-server-0              1/1     Running     0
1h
polaris-consul-consul-server-1              1/1     Running     0
1h
polaris-consul-consul-server-2              1/1     Running     0
1h
polaris-keycloak-0                          1/1     Running     2 (1h
ago)      1h
```

```
polaris-keycloak-1                          1/1     Running     0
1h
polaris-keycloak-db-0                       1/1     Running     0
1h
polaris-keycloak-db-1                       1/1     Running     0
1h
polaris-keycloak-db-2                       1/1     Running     0
1h
polaris-mongodb-0                           1/1     Running     0
1h
polaris-mongodb-1                           1/1     Running     0
1h
polaris-mongodb-2                           1/1     Running     0
1h
polaris-ui-66476dcf87-f6s8j                 1/1     Running     0
1h
polaris-ui-66476dcf87-ztjk7                 1/1     Running     0
1h
polaris-vault-0                             1/1     Running     0
1h
polaris-vault-1                             1/1     Running     0
1h
polaris-vault-2                             1/1     Running     0
1h
public-metrics-bfc4fc964-x4m79              1/1     Running     0
1h
storage-backend-metrics-7dbb88d4bc-g78cj    1/1     Running     0
1h
storage-provider-5969b5df5-hjvcm            1/1     Running     0
1h
storage-provider-5969b5df5-r79ld            1/1     Running     0
1h
task-service-5fc9dc8d99-4q4f4               1/1     Running     0
1h
task-service-5fc9dc8d99-815zl               1/1     Running     0
1h
task-service-task-purge-28485735-fdzkd      1/1     Running     0
12m
telegraf-ds-2rgm4                           1/1     Running     0
1h
telegraf-ds-4qp6r                           1/1     Running     0
1h
telegraf-ds-77frs                           1/1     Running     0
1h
telegraf-ds-bc725                           1/1     Running     0
1h
```

```
telegraf-ds-cvmxf                          1/1     Running     0
1h
telegraf-ds-tqzgj                          1/1     Running     0
1h
telegraf-rs-5wtd8                          1/1     Running     0
1h
telemetry-service-6747866474-5djnc         1/1     Running     0
1h
telemetry-service-6747866474-thb7r         1/1     Running     1 (1h
ago)      1h
tenancy-5669854fb6-gzdzf                   1/1     Running     0
1h
tenancy-5669854fb6-xvsm2                   1/1     Running     0
1h
traefik-8f55f7d5d-4lgfw                    1/1     Running     0
1h
traefik-8f55f7d5d-j4wt6                    1/1     Running     0
1h
traefik-8f55f7d5d-p6gcq                    1/1     Running     0
1h
trident-svc-7cb5bb4685-54cnq               1/1     Running     0
1h
trident-svc-7cb5bb4685-b28xh               1/1     Running     0
1h
vault-controller-777b9bbf88-b5bqt          1/1     Running     0
1h
vault-controller-777b9bbf88-fdfd8          1/1     Running     0
1h
```

4. (Optional) Watch the `acc-operator` logs to monitor progress:

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

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

5. When all the pods are running, verify that the installation was successful (`READY` is `True`) and get the initial setup password you'll 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  24.02.0-69
10.111.111.111  True
```

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

Set up ingress for load balancing

You can set up a Kubernetes ingress controller that manages external access to services. These procedures give setup examples for an ingress controller if you used the default of `ingressType: "Generic"` in the Astra Control Center custom resource (`astra_control_center.yaml`). You do not need to use this procedure if you specified `ingressType: "AccTraefik"` in the Astra Control Center custom resource (`astra_control_center.yaml`).

After Astra Control Center is deployed, you'll 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 some common ingress controller types.

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):

```
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'll 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.

   You can create a custom certificate after login.

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

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

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

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

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

Troubleshoot the installation

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

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
```

Alternative installation procedures

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

What's next

- (Optional) Depending on your environment, complete post-installation configuration steps.
- 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, enable authentication, manage storage, and add buckets.

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

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

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

- **Ensure 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
    ```

- **Ensure a routable FQDN**: The Astra FQDN you plan to use can be routed to the 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.
- **Obtain OpenShift permissions**: You'll need all necessary permissions and access to the Red Hat OpenShift Container Platform to perform the installation steps described.
- **Configure a cert manager**: 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.

- **Set up 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.

- **(ONTAP SAN driver only) Enable multipath**: If you are using an ONTAP SAN driver, be sure that multipath is enabled on all your Kubernetes clusters.

You should also consider the following:

- **Get access to the NetApp Astra Control image registry**:

  You have the option to obtain installation images and functionality enhancements for Astra Control, such as Astra Control Provisioner, from the NetApp image registry.

  1. Record your Astra Control account ID that you'll need to log in to the registry.

     You can see your account ID in the Astra Control Service web UI. Select the figure icon at the top right of the page, select **API access**, and write down your account ID.

  2. From the same page, select **Generate API token** and copy the API token string to the clipboard and save it in your editor.
  3. Log into the Astra Control registry:

  ```
  docker login cr.astra.netapp.io -u <account-id> -p <api-token>
  ```

- **Install a service mesh for secure communications**: It is strongly recommended that Astra Control host cluster communications channels be secured using a supported service mesh.

  > Integrating Astra Control Center with a service mesh can only be done during Astra Control Center installation and not independent of this process. Changing back from a meshed to an unmeshed environment is not supported.

  For Istio service mesh use, you'll need to do the following:

  - Add an `istio-injection:enabled` label to the Astra namespace prior to deploying Astra Control Center.
  - Use the `Generic` ingress setting and provide an alternative ingress for external load balancing.
  - For Red Hat OpenShift clusters, you'll need to define `NetworkAttachmentDefinition` on all associated Astra Control Center namespaces (`netapp-acc-operator`, `netapp-acc`, `netapp-monitoring` for application clusters, or any custom namespaces that have been substituted).

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

Steps

- Download and extract Astra Control Center
- Complete additional steps if you use a local registry
- Find the operator install page
- Install the operator
- Install Astra Control Center

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.

Download and extract Astra Control Center

Download the Astra Control Center images from one of the following locations:

- **Astra Control Service image registry**: Use this option if you don't use a local registry with the Astra Control Center images or if you prefer this method to the bundle download from the NetApp Support Site.
- **NetApp Support Site**: Use this option if you use a local registry with the Astra Control Center images.

Astra Control image registry

1. Log in to Astra Control Service.
2. On the Dashboard, select **Deploy a self-managed instance of Astra Control**.
3. Follow the instructions to log in to the Astra Control image registry, pull the Astra Control Center installation image, and extract the image.

NetApp Support Site

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

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

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

Complete additional steps if you use a local registry

If you are planning to push the Astra Control Center bundle to your local registry, you need to use the NetApp Astra kubectl command line plugin.

Install the NetApp Astra kubectl plugin

Complete these steps to install the most recent NetApp Astra kubectl command line plugin.

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:

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 registry

1. If you are planning to push the Astra Control Center bundle to your local registry, complete the appropriate step sequence for your container engine:

Docker

a. Change to the root directory of the tarball. You should see the `acc.manifest.bundle.yaml` file and these directories:

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

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/
kubectl-astra/
acc.manifest.bundle.yaml
```

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=24.02.0-69
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=24.02.0-69
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://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. Change the directory:

```
cd manifests
```

Find the operator install page

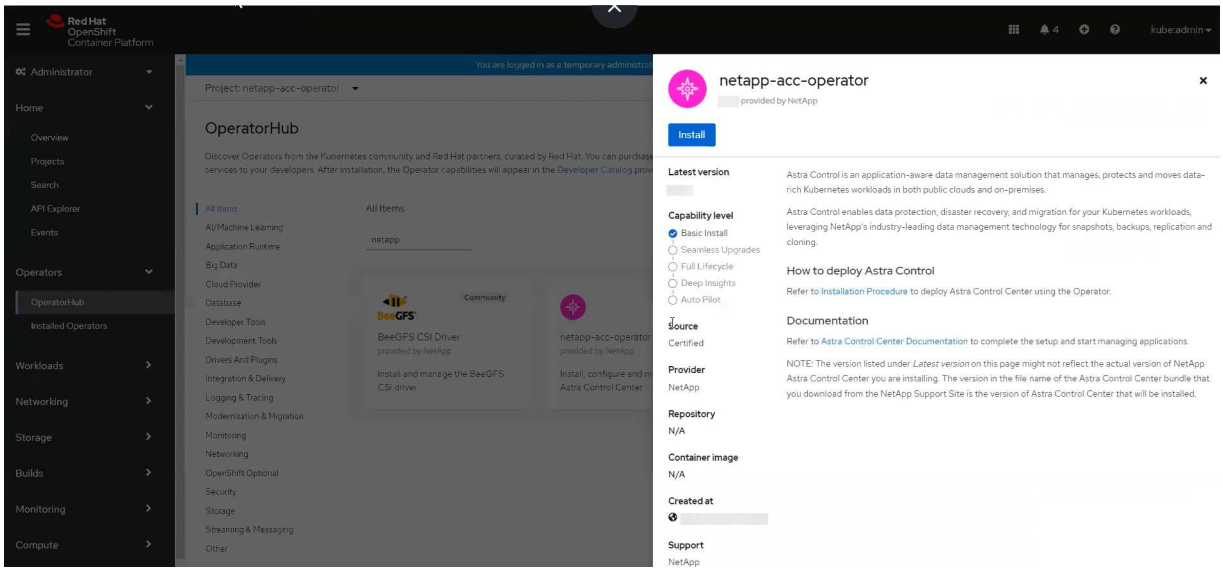
1. Complete one of the following procedures to access the operator install page:

Red Hat OpenShift web console

1. Log in to the OpenShift Container Platform UI.
2. From the side menu, select **Operators > OperatorHub**.

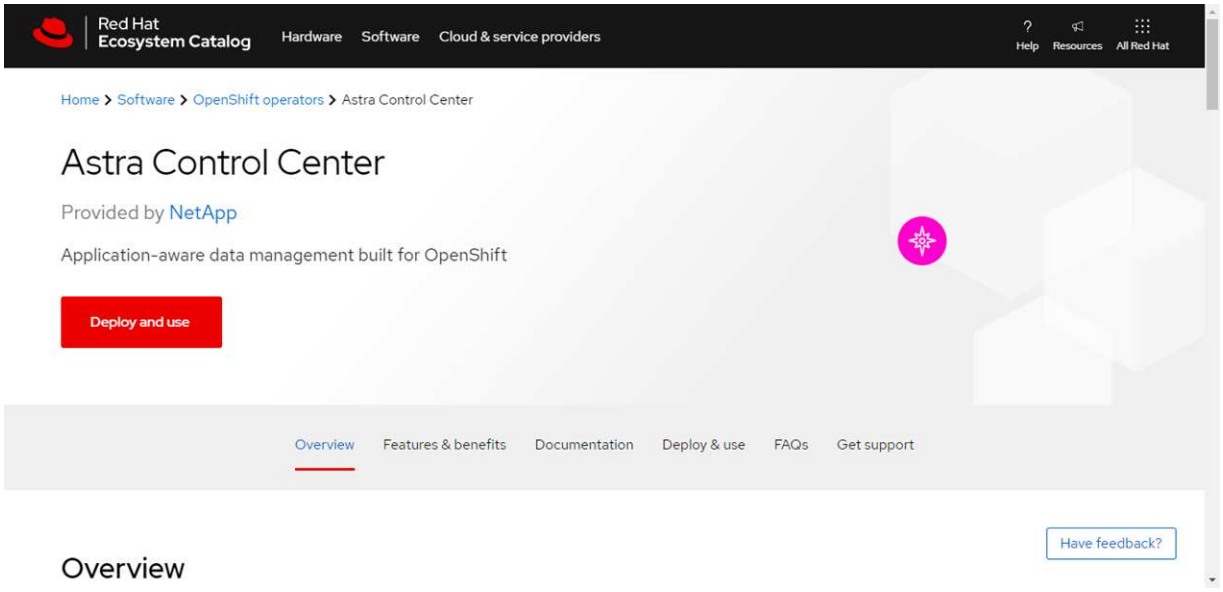
> You can upgrade only to the current version of Astra Control Center using this operator.

3. Search for `netapp-acc` and select the NetApp Astra Control Center operator.



Red Hat Ecosystem Catalog

1. Select the NetApp Astra Control Center operator.
2. 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.

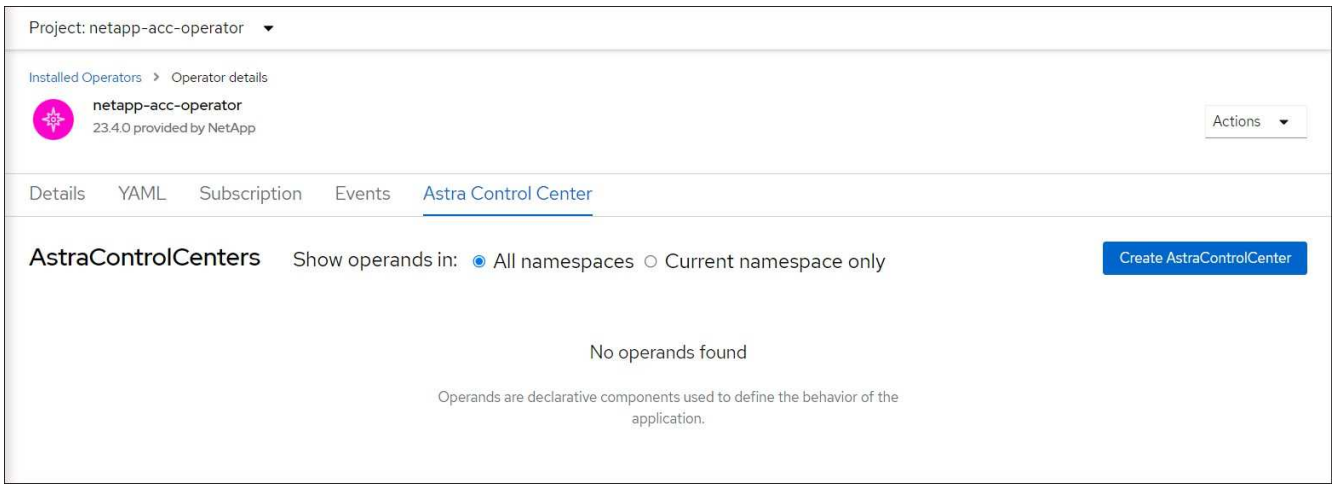
   c. Select **Install**.

   If you selected a manual approval strategy, you'll 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, 24.02.0-69.

   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 scale size of the installation.

> 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'll need to configure the ingress controller to expose Astra Control Center with a URL.

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

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

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

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

j. In **Image Registry**, use the default value unless you configured a local registry. For a local registry, replace this value with the local image registry path where you pushed the images in a previous step. Do not enter `http://` or `https://` in the address field.

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

> 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.

> 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]
```

> 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'll 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-premises 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'll 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 Amazon 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 4.11 through 4.13

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 specific resources in addition to the environment's resource requirements. Refer to Astra Control Center operational environment requirements.

> The AWS registry token expires in 12 hours, after which you'll 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, and create an AWS S3 bucket. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create an Elastic Container Registry (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.

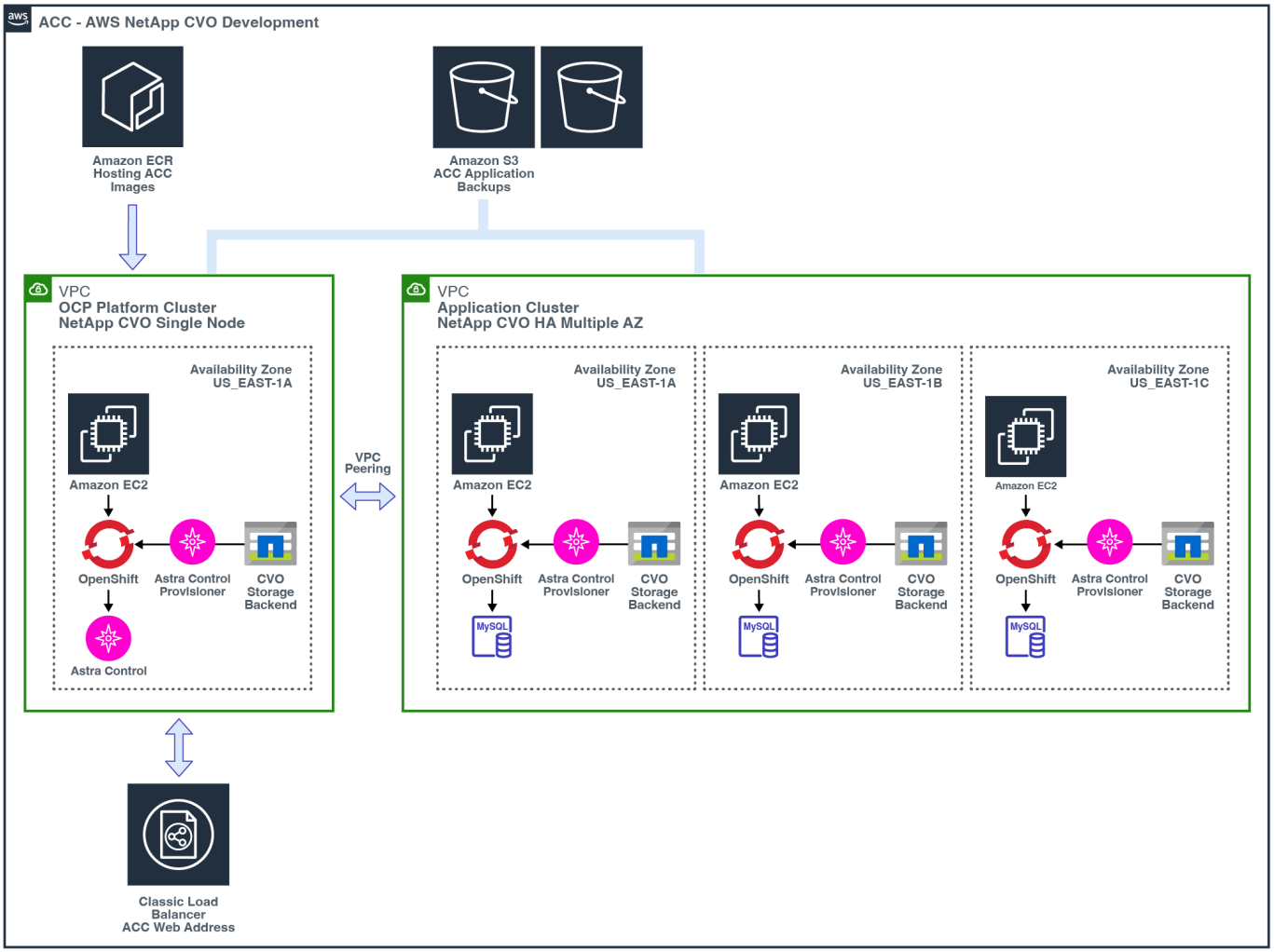
1. Create an AWS virtual network.
2. Review the EC2 compute instances. This can be a bare metal server or VMs in AWS.
3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in AWS to meet the Astra requirements. Refer to Astra Control Center requirements.
4. Create at least one AWS S3 bucket to store your backups.
5. (Optional) If you cannot access the NetApp image registry, do the following:
   a. Create an AWS Elastic Container Registry (ECR) to host all the Astra Control Center 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.

   b. Push the Astra Control Center 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 Control Provisioner 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 Control Provisioner 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'll 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) 4.11 through 4.13
- 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 specific resources in addition to the environment's resource requirements. Refer to Astra Control Center operational environment requirements.

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).

> 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, and create a Google Cloud Object Storage. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create a Google Container Registry 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. (Optional) If you cannot access the NetApp image registry, do the following:
    a. Create a Google Container Registry to host the Astra Control Center images.
    b. Set up Google Container Registry access for Docker push/pull for all the Astra Control Center images.

Example: Astra Control Center 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=acc.manifest.bundle.yaml
GCP_CR_REGISTRY=<target GCP image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 < acc.manifest.bundle.yaml
```

7. 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 Astra Control Provisioner 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 Control Provisioner 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'll 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) 4.11 through 4.13
- 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 specific resources in addition to the environment's resource requirements. Refer

to Astra Control Center operational environment requirements.

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, and create an Azure Blob container. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create an Azure Container Registry (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'll 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. (Optional) If you cannot access the NetApp image registry, do the following:
   a. Create an Azure Container Registry (ACR) to host the Astra Control Center images.
   b. Set up ACR access for Docker push/pull for all the Astra Control Center images.
   c. Push the Astra Control Center images to this registry using the following script:

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

   Example:

```
manifestfile=acc.manifest.bundle.yaml
AZ_ACR_REGISTRY=<target Azure ACR image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

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 $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. 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

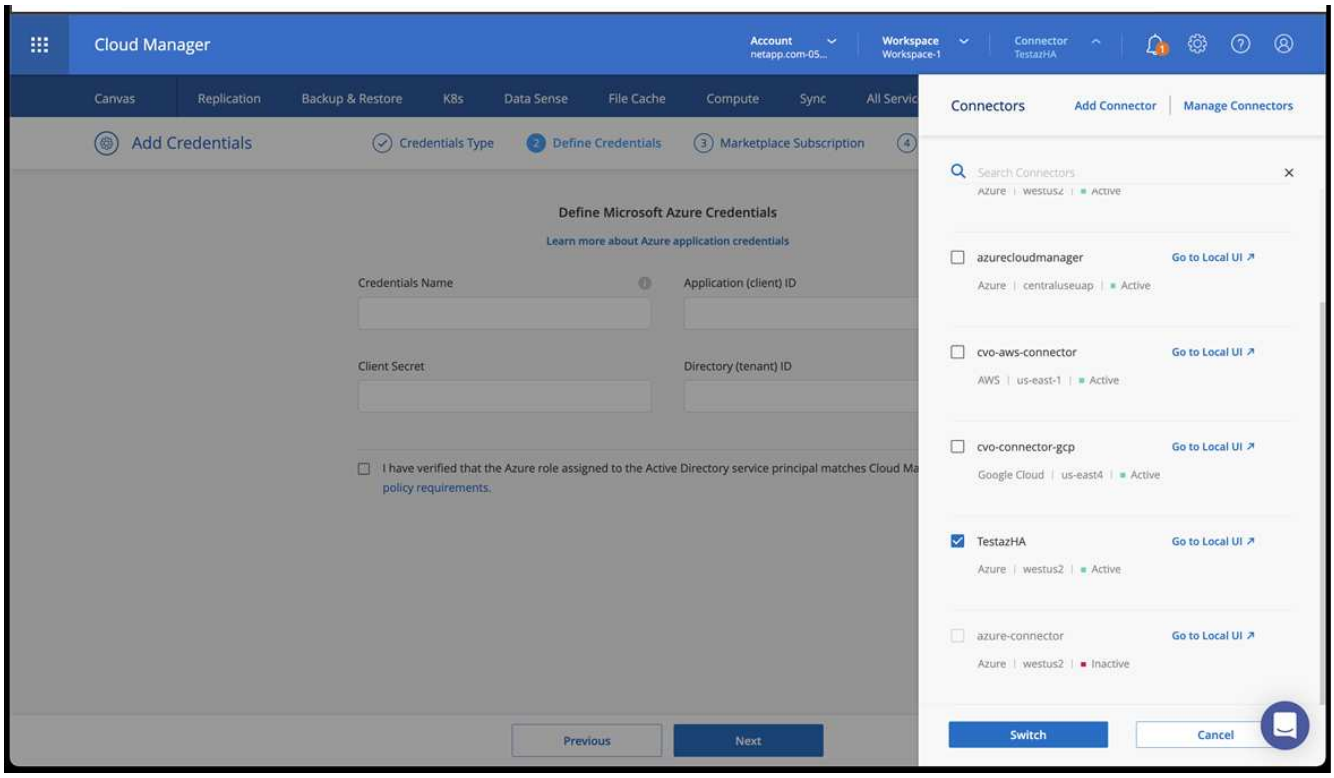
Steps

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.

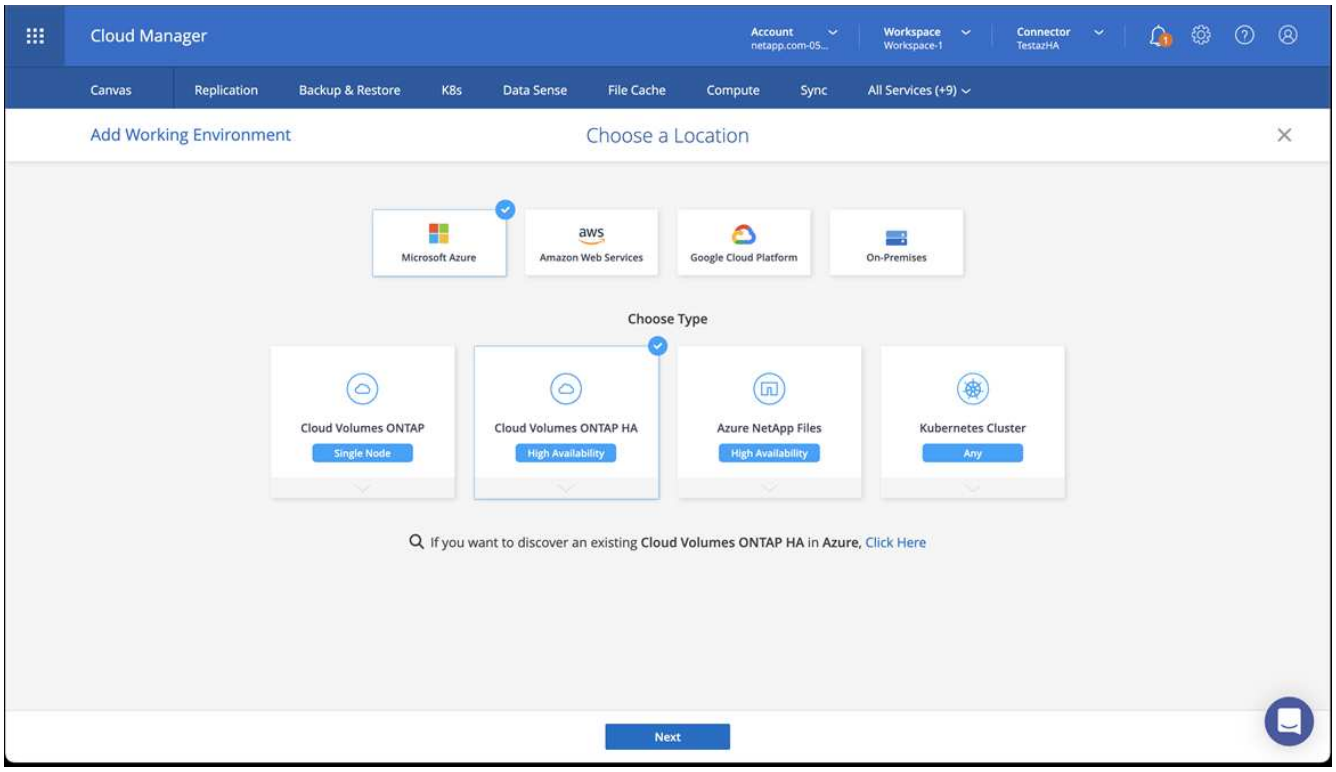
   See Creating a connector in Azure from BlueXPr.

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



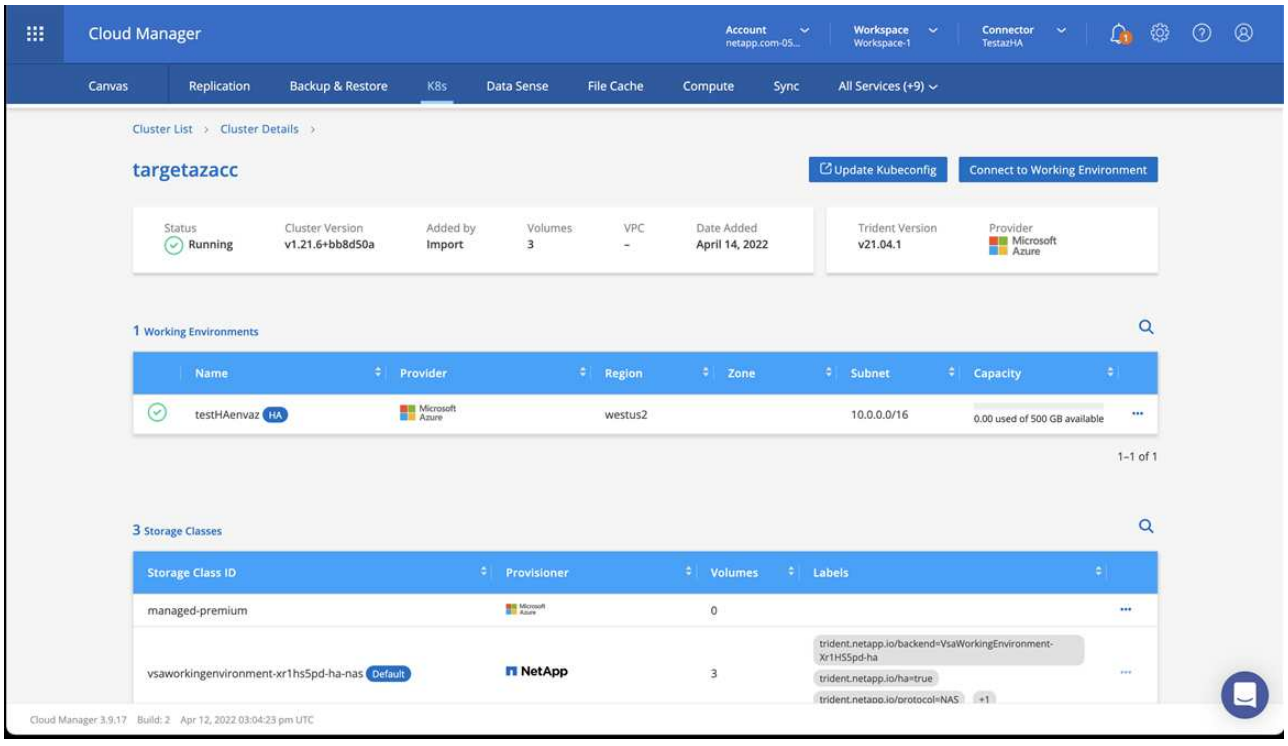
4. Create a working environment for your cloud environment.

   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 Control Provisioner 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 Control Provisioner 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]
```

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. For production use, you should 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>
```

CRD:

```
#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>
```

Response:

```
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:

This example uses the the `dnsNames` property to specify the Astra Control Center DNS address. Astra Control Center does not support using the Common Name (CN) property to specify the DNS address.

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  <strong>name: <certificate-name></strong>
  namespace: <ACC-deployment-namespace>
spec:
  <strong>secretName: <certificate-secret-name></strong>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  <strong>- <astra.dnsname.example.com></strong> #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>
```

Response:

```
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 TLS stores CRD to point to your new certificate secret name using the following command and example, replacing the placeholder values in brackets <> with appropriate information

```
kubectl edit tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD:

```
...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>
```

8. 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.io -n <ACC-deployment-namespace>
```

CRD:

```
...
 tls:
    secretName: <certificate-secret-name>
```

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

Set up Astra Control Center

Add a license for Astra Control Center

When you install Astra Control Center, an embedded evaluation license is already installed. If you are evaluating Astra Control Center, you can skip this step.

You can add a new license using the Astra Control UI or Astra Control API.

Astra Control Center licenses measure CPU resources using Kubernetes CPU units and account for the CPU resources assigned to the worker nodes of all the managed Kubernetes clusters. Licenses are based on vCPU usage. For more information on how licenses are calculated, refer to Licensing.

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.

To update an existing evaluation or full license, refer to Update an existing license.

Before you begin

- Access to a newly installed Astra Control Center instance.
- Administrator role permissions.
- A NetApp License File (NLF).

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 and are not sending data to AutoSupport, be sure that you store your account ID to avoid data loss in the event of Astra Control Center failure.

Enable Astra Control Provisioner

Astra Trident versions 23.10 and later include the option to use Astra Control Provisioner, which enables licensed Astra Control users to access advanced storage provisioning functionality. Astra Control Provisioner provides this extended functionality in addition to standard Astra Trident CSI-based functionality.

In coming Astra Control updates, Astra Control Provisioner will replace Astra Trident as storage provisioner and orchestrator and be mandatory for Astra Control use. Because of this, it's strongly recommended that Astra Control users enable Astra Control Provisioner. Astra Trident will continue to remain open source and be released, maintained, supported, and updated with new CSI and other features from NetApp.

About this task

You should follow this procedure if you are a licensed Astra Control Center user and you are looking to use Astra Control Provisioner functionality. You should also follow this procedure if you are an Astra Trident user and want to use the additional functionality that Astra Control Provisioner provides without also using Astra Control.

For each case, the provisioner functionality is not enabled by default in Astra Trident 24.02 and must be enabled.

Before you begin

If you are enabling Astra Control Provisioner, do the following first:

Astra Control Provisioners users with Astra Control Center

- **Obtain an Astra Control Center license**: You'll need an Astra Control Center license to enable Astra Control Provisioner and access the functionality it provides.
- **Install or upgrade to Astra Control Center 23.10 or later**: You'll need the latest Astra Control Center version (24.02) if you are planning to use the latest Astra Control Provisioner functionality (24.02) with Astra Control.
- **Confirm that your cluster has an AMD64 system architecture**: The Astra Control Provisioner image is provided in both AMD64 and ARM64 CPU architectures, but only AMD64 is supported by Astra Control Center.
- **Get an Astra Control Service account for registry access**: If you intend to use the Astra Control registry rather than the NetApp Support Site to download the Astra Control Provisioner image, complete the registration for an Astra Control Service account. After you complete and submit the form and create a BlueXP account, you'll receive an Astra Control Service welcome email.
- **If you have Astra Trident installed, confirm that its version is within a four-release window**: You can perform a direct upgrade to Astra Trident 24.02 with Astra Control Provisioner if your Astra Trident is within a four-release window of version 24.02. For example, you can directly upgrade from Astra Trident 23.04 to 24.02.

Astra Control Provisioner only users

- **Obtain an Astra Control Center license**: You'll need an Astra Control Center license to enable Astra Control Provisioner and access the functionality it provides.
- **If you have Astra Trident installed, confirm that its version is within a four-release window**: You can perform a direct upgrade to Astra Trident 24.02 with Astra Control Provisioner if your Astra Trident is within a four-release window of version 24.02. For example, you can directly upgrade from Astra Trident 23.04 to 24.02.
- **Get an Astra Control Service account for registry access**: You'll need access to the registry to download Astra Control Provisioner images. To get started, complete the registration for an Astra Control Service account. After you complete and submit the form and create a BlueXP account, you'll receive an Astra Control Service welcome email.

(Step 1) Get the Astra Control Provisioner image

Astra Control Center users can get the Astra Control Provisioner image using either the Astra Control registry or NetApp Support Site method. Astra Trident users wanting to use Astra Control Provisioner without Astra Control should use the registry method.

Astra Control image registry

> You can use Podman instead of Docker for the commands in this procedure. If you are using a Windows environment, PowerShell is recommended.

1. Access the NetApp Astra Control image registry:
   a. Log on to the Astra Control Service web UI and select the figure icon at the top right of the page.
   b. Select **API access**.
   c. Write down your account ID.
   d. From the same page, select **Generate API token** and copy the API token string to the clipboard and save it in your editor.
   e. Log into the Astra Control registry using your preferred method:

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (Custom registries only) Follow these steps to move the image to your custom registry. If you aren't using a registry, follow the Trident operator steps in the next section.
   a. Pull the Astra Control Provisioner image from the registry:

> The image pulled will not support multiple platforms and will only support the same platform as the host that pulled the image, such as Linux AMD64.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform <cluster platform>
```

   Example:

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform linux/amd64
```

   b. Tag the image:

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0
```

   c. Push the image to your custom registry:

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

> You can use Crane copy as an alternative to running these Docker commands:
> `crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0`

NetApp Support Site

1. Download the Astra Control Provisioner bundle (`trident-acp-[version].tar`) from the Astra Control Center downloads page.
2. (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 trident-acp-[version] tar bundle.

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

```
openssl dgst -sha256 -verify certs/AstraControlCenterDockerImages-public.pub -signature certs/trident-acp-[version].tar.sig trident-acp-[version].tar
```

3. Load the Astra Control Provisioner image:

```
docker load < trident-acp-24.02.0.tar
```

   Response:

```
Loaded image: trident-acp:24.02.0-linux-amd64
```

4. Tag the image:

```
docker tag trident-acp:24.02.0-linux-amd64 <my_custom_registry>/trident-acp:24.02.0
```

5. Push the image to your custom registry:

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

(Step 2) Enable Astra Control Provisioner in Astra Trident

Determine if the original installation method used an operator (either manually or with Helm) or tridentctl and complete the appropriate steps according to your original method.

Astra Trident operator

1. Download the Astra Trident installer and extract it.
2. Complete these steps if you have not yet installed Astra Trident or if you removed the operator from your original Astra Trident deployment:
   a. Create the CRD:

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.y
aml
```

   b. Create the trident namespace (`kubectl create namespace trident`) or confirm that the trident namespace still exists (`kubectl get all -n trident`). If the namespace has been removed, create it again.
3. Update Astra Trident to 24.02.0:

> For clusters running Kubernetes 1.24 or earlier, use `bundle_pre_1_25.yaml`. For clusters running Kubernetes 1.25 or later, use `bundle_post_1_25.yaml`.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

4. Verify Astra Trident is running:

```
kubectl get torc -n trident
```

   Response:

```
NAME      AGE
trident   21m
```

5. If you have a registry that uses secrets, create a secret to use to pull the Astra Control Provisioner image:

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

6. Edit the TridentOrchestrator CR and make the following edits:

```
kubectl edit torc trident -n trident
```

a. Set a custom registry location for the Astra Trident image or pull it from the Astra Control registry (`tridentImage: <my_custom_registry>/trident:24.02.0` or `tridentImage: netapp/trident:24.02.0`).

b. Enable Astra Control Provisioner (`enableACP: true`).

c. Set the custom registry location for the Astra Control Provisioner image or pull it from the Astra Control registry (`acpImage: <my_custom_registry>/trident-acp:24.02.0` or `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`).

d. If you established image pull secrets earlier in this procedure, you can set them here (`imagePullSecrets: - <secret_name>`). Use the same name secret name you established in the previous steps.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

7. Save and exit the file. The deployment process will begin automatically.
8. Verify the operator, deployment, and replicasets are created.

```
kubectl get all -n trident
```

> There should only be **one instance** of the operator in a Kubernetes cluster. Do not create multiple deployments of the Astra Trident operator.

9. Verify the `trident-acp` container is running and that `acpVersion` is `24.02.0` with a status of `Installed`:

```
kubectl get torc -o yaml
```

Response:

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl

1. Download the Astra Trident installer and extract it.
2. If you have an existing Astra Trident, uninstall it from the cluster that hosts it.
3. Install Astra Trident with Astra Control Provisioner enabled (`--enable-acp=true`):

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

4. Confirm that Astra Control Provisioner has been enabled:

```
./tridentctl -n trident version
```

Response:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. | +----------------
+----------------+-------------+
```

Helm

1. If you have Astra Trident 23.07.1 or earlier installed, uninstall the operator and other components.
2. If your Kubernetes cluster is running 1.24 or earlier, delete psp:

```
kubectl delete psp tridentoperatorpod
```

3. Add the Astra Trident Helm repository:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

4. Update the Helm chart:

```
helm repo update netapp-trident
```

Response:

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

5. List the images:

```
./tridentctl images -n trident
```

Response:

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-driver-
registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

6. Ensure that trident-operator 24.02.0 is available:

```
helm search repo netapp-trident/trident-operator --versions
```

Response:

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

7. Use `helm install` and run one of the following options that include these settings:
   - A name for your deployment location
   - The Astra Trident version
   - The name of the Astra Control Provisioner image
   - The flag to enable the provisioner
   - (Optional) A local registry path. If you are using a local registry, your Trident images can be located in one registry or different registries, but all CSI images must be located in the same registry.
   - The Trident namespace

Options

- Images without a registry

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-acp:24.02.0
--set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- Images in one or more registries

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

You can use `helm list` to review installation details such as name, namespace, chart, status, app version, and revision number.

If you have any issues deploying Trident using Helm, run this command to fully uninstall Astra Trident:

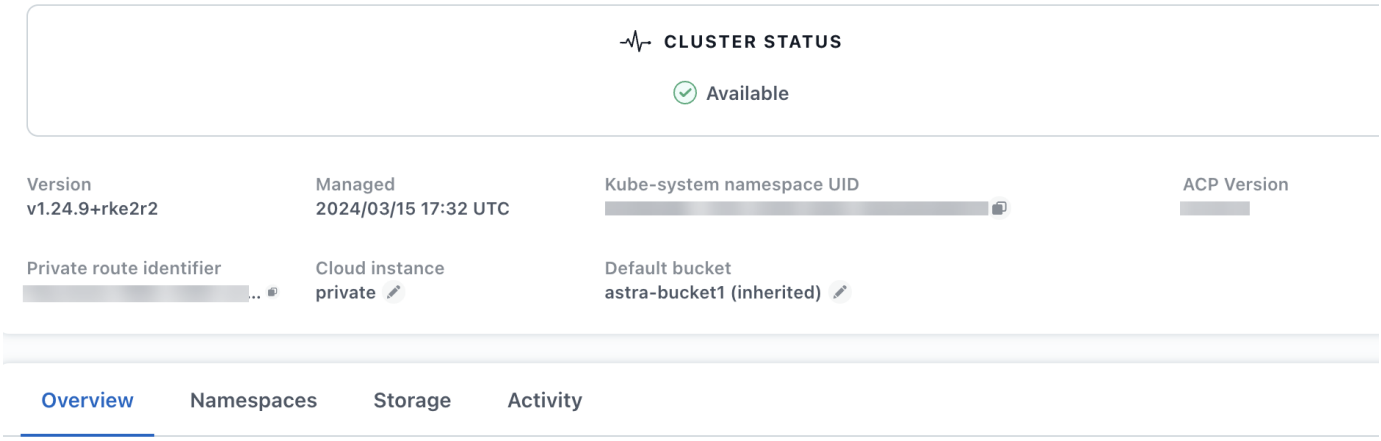
```
./tridentctl uninstall -n trident
```

Do not completely remove Astra Trident CRDs as part of your uninstall before attempting to enable Astra Control Provisioner again.

Result

Astra Control Provisioner functionality is enabled and you can use any features available for the version you are running.

(For Astra Control Center users only) After Astra Control Provisioner is installed, the cluster hosting the provisioner in the Astra Control Center UI will show an `ACP version` rather than `Trident version` field and current installed version number.



For more information

- Astra Trident upgrades documentation

Prepare your environment for cluster management using Astra Control

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

Astra Control allows you to add clusters managed by custom resource (CR) or kubeconfig, depending on your environment and preferences.

Before you begin

- **Meet environmental prerequisites**: Your environment meets operational environment requirements for Astra Control Center.
- **Configure worker nodes**: Ensure that you configure the worker nodes in your cluster with the appropriate storage drivers so that the pods can interact with the backend storage.

- **Enable PSA restrictions**: If your cluster has pod security admission enforcement enabled, which is standard for Kubernetes 1.25 and later clusters, you need to enable PSA restrictions on these namespaces:
  - `netapp-acc-operator` namespace:

    ```
    kubectl label --overwrite ns netapp-acc-operator pod-
    security.kubernetes.io/enforce=privileged
    ```

  - `netapp monitoring` namespace:

    ```
    kubectl label --overwrite ns netapp-monitoring pod-
    security.kubernetes.io/enforce=privileged
    ```

- **ONTAP credentials**: You need ONTAP credentials and a superuser and user ID set on the backing ONTAP system to back up and restore apps with Astra Control Center.

  Run the following commands in the ONTAP command line:

  ```
  export-policy rule modify -vserver <storage virtual machine name>
  -policyname <policy name> -ruleindex 1 -superuser sys
  export-policy rule modify -vserver <storage virtual machine name>
  -policyname <policy name> -ruleindex 1 -anon 65534
  ```

- **kubeconfig-managed cluster requirements**: These requirement are specific for app clusters managed by kubeconfig.
  - **Make kubeconfig accessible**: You have access to the default cluster kubeconfig that you configured during installation.
  - **Certificate Authority considerations**: If you are adding the cluster using a kubeconfig file that references a private Certificate Authority (CA), add the following line to the `cluster` section of the kubeconfig file. This enables Astra Control to add the cluster:

    ```
    insecure-skip-tls-verify: true
    ```

  - **Rancher only**: When managing application clusters in a Rancher environment, modify the application cluster's default context in the kubeconfig file provided by Rancher to use a control plane context instead of the Rancher API server context. This reduces load on the Rancher API server and improves performance.
- **Astra Control Provisioner requirements**: You should have a properly configured Astra Control Provisioner, including its Astra Trident components, to manage clusters.
  - **Review Astra Trident environment requirements**: Prior to installing or upgrading Astra Control Provisioner, review the supported frontends, backends, and host configurations.
  - **Enable Astra Control Provisioner functionality**: It's highly recommended that you install Astra Trident 23.10 or later and enable Astra Control Provisioner advanced storage functionality. In coming

releases, Astra Control will not support Astra Trident if the Astra Control Provisioner is not also enabled.

- **Configure a storage backend**: At least one storage backend must be configured in Astra Trident on the cluster.
- **Configure a storage class**: At least one storage class must be configured in Astra Trident on the cluster. If a default storage class is configured, ensure that it is the **only** storage class that has the default annotation.
- **Configure a volume snapshot controller and install a volume snapshot class**: Install a volume snapshot controller so that snapshots can be created in Astra Control. Create at least one `VolumeSnapshotClass` using Astra Trident.

Run eligibility checks

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

Steps

1. Determine the Astra Trident version you are running:

   ```
   kubectl get tridentversion -n trident
   ```

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

   ```
   NAME      VERSION
   trident   24.02.0
   ```

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

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

2. Do one of the following:
   - If you are running Astra Trident 23.01 or earlier, use these instructions to upgrade to a more recent version of Astra Trident before upgrading to the Astra Control Provisioner. You can perform a direct upgrade to Astra Control Provisioner 24.02 if your Astra Trident is within a four-release window of version 24.02. For example, you can directly upgrade from Astra Trident 23.04 to Astra Control Provisioner 24.02.
   - If you are running Astra Trident 23.10 or later, verify that Astra Control Provisioner has been enabled. Astra Control Provisioner will not work with releases of Astra Control Center earlier than 23.10. Upgrade your Astra Control Provisioner so that it has the same version as the Astra Control Center you are upgrading to access the latest functionality.
3. Ensure that all pods (including `trident-acp`) are running:

   ```
   kubectl get pods -n trident
   ```

4. Determine if the storage classes are using the supported Astra Trident drivers. The provisioner name

should be `csi.trident.netapp.io`. See the following example:

```
kubectl get sc
```

Sample response:

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
ontap-gold (default)  csi.trident.netapp.io  Delete          Immediate
true                    5d23h
```

Create a cluster role kubeconfig

For clusters that are managed using kubeconfig, you can optionally create a limited permission or expanded permission administrator role for Astra Control Center. This is not a required procedure for Astra Control Center setup as you already configured a kubeconfig as part of the installation process.

This procedure helps you to create a separate kubeconfig if either of the following scenarios applies to your environment:

- You want to limit Astra Control permissions on the clusters it manages
- You use multiple contexts and cannot use the default Astra Control kubeconfig configured during installation or a limited role with a single context won't work in your environment

Before you begin

Ensure that you have the following for the cluster you intend to manage before completing the procedure steps:

- kubectl v1.23 or later installed
- kubectl access to the cluster that you intend to add and manage with Astra Control Center

> For this procedure, you do not need kubectl access to the cluster that is running Astra Control Center.

- An active kubeconfig for the cluster you intend to manage with cluster admin rights for the active context

Steps

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

```
<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. Create one of the following cluster roles with sufficient permissions for a cluster to be managed by Astra Control:

Limited cluster role

This role contains the minimum permissions necessary for a cluster to be managed by Astra Control:

1. Create a `ClusterRole` file called, for example, `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
  - horizontalpodautoscalers
  - ingresses
  - jobs
  - namespaces
  - networkpolicies
  - persistentvolumeclaims
  - poddisruptionbudgets
  - pods
  - podtemplates
  - replicasets
  - replicationcontrollers
  - replicationcontrollers/scale
  - rolebindings
  - roles
  - secrets
  - serviceaccounts
  - services
  - statefulsets
  - tridentmirrorrelationships
  - tridentsnapshotinfos
  - volumesnapshots
  - volumesnapshotcontents
  verbs:
  - delete

# Watch resources
# Necessary to monitor progress
- apiGroups:
  - ""
  resources:
  - pods
  - replicationcontrollers
  - replicationcontrollers/scale
  verbs:
  - watch

# Update resources
- apiGroups:
  - ""
  - build.openshift.io
  - image.openshift.io
  resources:
  - builds/details
  - replicationcontrollers
  - replicationcontrollers/scale
  - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

2. (For OpenShift clusters only) Append the following at the end of the `astra-admin-account.yaml` file:

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

3. Apply the cluster role:

```
kubectl apply -f astra-admin-account.yaml
```

Expanded cluster role

This role contains expanded permissions for a cluster to be managed by Astra Control. You might use this role if you use multiple contexts and cannot use the default Astra Control kubeconfig configured during installation or a limited role with a single context won't work in your environment:

> The following `ClusterRole` steps are a general Kubernetes example. Refer to the documentation for your Kubernetes distribution for instructions specific to your environment.

1. Create a `ClusterRole` file called, for example, `astra-admin-account.yaml`.

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

2. Apply the cluster role:

```
kubectl apply -f astra-admin-account.yaml
```

3. Create the cluster role binding for the cluster role to the service account:
   a. Create a `ClusterRoleBinding` file called `astracontrol-clusterrolebinding.yaml`.

```
<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: astra-admin-account
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

   b. Apply the cluster role binding:

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

4. Create and apply the token secret:
   a. Create a token secret file called `secret-astracontrol-service-account.yaml`.

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

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. Apply the token secret:

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

5. Add the token secret to the service account by adding its name to the `secrets` array (the last line in the following example):

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. 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-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

The indices for each element in the `secrets` array begin with 0. In the above example, the index for `astracontrol-service-account-dockercfg-48xhx` would be 0 and the index for `secret-astracontrol-service-account` would be 1. In your output, make note of the index number for the service account secret. You'll need this index number in the next step.

7. Generate the kubeconfig as follows:
   a. Create a `create-kubeconfig.sh` file.
   b. 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
```

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

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

8. (Optional) Rename the kubeconfig to a meaningful name for your cluster.

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(Tech preview) Install Astra Connector for managed clusters

Clusters managed by Astra Control Center use Astra Connector to enable communication between the managed cluster and Astra Control Center. You need to install Astra Connector on all clusters that you want to manage.

Install Astra Connector

You install Astra Connector using Kubernetes commands and Custom Resource (CR) files.

About this task

- When you perform these steps, execute these commands on the cluster that you want to manage with Astra Control.
- If you are using a bastion host, issue these commands from the command line of the bastion host.

Before you begin

- You need access to the cluster you want to manage with Astra Control.
- You need Kubernetes administrator permissions to install the Astra Connector operator on the cluster.

If the cluster is configured with pod security admission enforcement, which is the default for Kubernetes 1.25 and later clusters, you need to enable PSA restrictions on the appropriate namespaces. Refer to Prepare your environment for cluster management using Astra Control for instructions.

Steps

1. Install the Astra Connector operator on the cluster you want to manage with Astra Control. When you run this command, the namespace `astra-connector-operator` is created and the configuration is applied to the namespace:

```
kubectl apply -f https://github.com/NetApp/astra-connector-
operator/releases/download/24.02.0-
202403151353/astraconnector_operator.yaml
```

2. Verify that the operator is installed and ready:

```
kubectl get all -n astra-connector-operator
```

3. Get an API token from Astra Control. Refer to the Astra Automation documentation for instructions.
4. Create a secret using the token. Replace <API_TOKEN> with the token you received from Astra Control:

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. Create a Docker secret to use to pull the Astra Connector image. Replace values in brackets <> with information from your environment:

You can find the <ASTRA_CONTROL_ACCOUNT_ID> in the Astra Control web UI. In the web UI, select the figure icon at the top right of the page and select **API access**.

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. Create the Astra Connector CR file and name it `astra-connector-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <ASTRA_CONTROL_ACCOUNT_ID>: Obtained from the Astra Control web UI during the preceding step.
   - <CLUSTER_NAME>: The name that this cluster should be assigned in Astra Control.

- <ASTRA_CONTROL_URL>: The web UI URL of Astra Control. For example:

```
https://astra.control.url
```

```
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. After you populate the `astra-connector-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. Verify that the Astra Connector is fully deployed:

```
kubectl get all -n astra-connector
```

9. Verify that the cluster is registered with Astra Control:

```
kubectl get astraconnectors.astra.netapp.io -A
```

You should see output similar to the following:

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e   Registered with Astra
```

10. Verify that the cluster appears in the list of managed clusters on the **Clusters** page of the Astra Control web UI.

Add a 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.

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.

Before you begin

- Before you add a cluster, review and perform the necessary prerequisite tasks.
- If you are using an ONTAP SAN driver, be sure that multipath is enabled on all your Kubernetes clusters.

Steps

1. Navigate from either the Dashboard or the Clusters menu:
   - From **Dashboard** in the Resource Summary, select **Add** from the Clusters pane.
   - In the left navigation area, select **Clusters** and then select **Add Cluster** from the Clusters page.
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. Refer to Kubernetes documentation for information about creating `kubeconfig` files. If you created a kubeconfig for a limited cluster role using this process, be sure to upload or paste that kubeconfig in this step.

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

   You should select a storage class that is configured in Astra Control Provisioner and backed by ONTAP storage.

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

Result

The cluster enters **Discovering** state and then changes to **Healthy**. You are now managing the cluster with 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'll need to look into the monitoring operator logs to debug the problem.

Enable authentication on an ONTAP storage backend

Astra Control Center offers two modes of authenticating an ONTAP backend:

- **Credential-based authentication**: The username and password to an ONTAP user with the required permissions. You should use a pre-defined security login role, such as admin or vsadmin to ensure maximum compatibility with ONTAP versions.
- **Certificate-based authentication**: Astra Control Center can also communicate with an ONTAP cluster using a certificate installed on the backend. You should use the client certificate, key, and the trusted CA certificate if used (recommended).

You can later update existing backends to move from one type of authentication to another method. Only one authentication method is supported at a time.

Enable credential-based authentication

Astra Control Center requires the credentials to a cluster-scoped `admin` to communicate with the ONTAP backend. You should use standard, pre-defined roles such as `admin`. This ensures forward compatibility with future ONTAP releases that might expose feature APIs to be used by future Astra Control Center releases.

> A custom security login role can be created and used with Astra Control Center, but is not recommended.

A sample backend definition looks like this:

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

The backend definition is the only place the credentials are stored in plain text. The creation or update of a backend is the only step that requires knowledge of the credentials. As such, it is an admin-only operation, to

be performed by the Kubernetes or storage administrator.

Enable certificate-based authentication

Astra Control Center can use certificates to communicate with new and existing ONTAP backends. You should enter the following information in the backend definition.

- `clientCertificate`: Client certificate.
- `clientPrivateKey`: Associated private key.
- `trustedCACertificate`: Trusted CA certificate. If using a trusted CA, this parameter must be provided. This can be ignored if no trusted CA is used.

You can use one of the following types of certificates:

- Self-signed certificate
- Third-party certificate

Enable authentication with a self-signed certificate

A typical workflow involves the following steps.

Steps

1. Generate a client certificate and key. When generating, set the Common Name (CN) to the ONTAP user to authenticate as.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. Install the client certificate of type `client-ca` and key on the ONTAP cluster.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. Confirm that the ONTAP security login role supports the certificate authentication method.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. Test authentication using the generated certificate. Replace <ONTAP Management LIF> and <vserver name> with the Management LIF IP and SVM name. You must ensure the LIF has its service policy set to `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. Using the values obtained from the previous step, add the storage backend in the Astra Control Center UI.

Enable authentication with a third-party certificate

If you have a third-party certificate, you can set up certificate-based authentication with these steps.

Steps

1. Generate the private key and CSR:

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. Pass the CSR to the Windows CA (third-party CA) and issue the signed certificate.
3. Download the signed certificate and name it `ontap_signed_cert.crt'
4. Export the root certificate from Windows CA (third-party CA).
5. Name this file `ca_root.crt`

   You now have the following three files:

   - **Private key**: `ontap_signed_request.key` (This is the corresponding key for the server certificate in ONTAP. It is needed while installing the server certificate.)
   - **Signed certificate**: `ontap_signed_cert.crt` (This is also called the *server certificate* in ONTAP.)
   - **Root CA certificate**: `ca_root.crt` (This is also called the *server-ca certificate* in ONTAP.)

6. Install these certificates in ONTAP. Generate and install `server` and `server-ca` certificates on ONTAP.

Expand for sample.yaml

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. Create the client certificate for the same host for passwordless communication. Astra Control Center uses this process to communicate with ONTAP.
8. Generate and install the client certificates on ONTAP:

Expand for sample.yaml

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate's CA and serial number for reference:

CA:
serial:
The certificate's generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```


9. Add the storage backend in the Astra Control Center UI and provide the following values:
   - **Client Certificate**: ontap_test_client.pem
   - **Private Key**: ontap_test_client.key
   - **Trusted CA Certificate**: ontap_signed_cert.crt

Add a storage backend

After you set up the credentials or certificate authentication information, you can add an existing ONTAP storage backend to Astra Control Center to manage its resources.

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.

Adding and managing ONTAP storage backends in Astra Control Center is optional when using NetApp SnapMirror technology if you have enabled Astra Control Provisioner.

Steps

1. From the Dashboard in the left-navigation area, select **Backends**.
2. Select **Add**.
3. In the Use Existing section of the Add storage backend page, select **ONTAP**.
4. Select one of the following:
   - **Use administrator credentials**: Enter the ONTAP cluster management IP address and admin credentials. The credentials must be cluster-wide credentials.

     The user whose credentials you enter here must have the `ontapi` user login access method enabled within ONTAP System Manager on the ONTAP cluster. If you plan to use SnapMirror replication, apply user credentials with the "admin" role, which has the access methods `ontapi` and `http`, on both source and destination ONTAP clusters. Refer to Manage User Accounts in ONTAP documentation for more information.

   - **Use a certificate**: Upload the certificate `.pem` file, the certificate key `.key` file, and optionally the certificate authority file.
5. Select **Next**.
6. Confirm the backend details and select **Manage**.

Result

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

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

Add a bucket

You can add a bucket using the Astra Control UI or Astra Control API. 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.

You don't need a bucket in Astra Control if you are cloning your application configuration and persistent storage to the same cluster. Application snapshots functionality does not require a bucket.

Before you begin

- Ensure you have a bucket that is reachable from your clusters managed by Astra Control Center.
- Ensure you have credentials for the bucket.
- Ensure the bucket is one of the following types:
  - NetApp ONTAP S3
  - NetApp StorageGRID S3
  - Microsoft Azure
  - Generic S3

Amazon Web Services (AWS) and Google Cloud Platform (GCP) use the Generic S3 bucket type.

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.

Steps

1. In the left navigation area, select **Buckets**.
2. Select **Add**.
3. 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.

4. 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.

5. Enter the name or IP address of the S3 endpoint.
6. Under **Select Credentials**, choose either the **Add** or **Use existing** tab.
   - If you chose **Add**:
     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.
   - If you chose **Use existing**:
     a. Select the existing credentials you want to use with the bucket.
7. Select `Add`.

   When you add a bucket, Astra Control marks one bucket with the default bucket indicator. The first bucket that you create becomes the default bucket. As you add buckets, you can later decide to set another default bucket.

Copyright information

Trademark information