NetApp

Get started

Astra Control Center

NetApp
August 31, 2022

Table of Contents

Get started

Astra Control Center requirements

Get started by verifying the readiness of your operational environment, application clusters, applications, licenses, and web browser.

- Operational environment requirements
- Supported storage backends
- Application cluster requirements
- Application management requirements
- Replication prerequisites
- Access to the internet
- License
- Ingress for on-premises Kubernetes clusters
- Networking requirements
- Supported web browsers

Operational environment requirements

Astra Control Center has been validated on the following types of operational environments:

- Google Anthos 1.10 or 1.11
- Kubernetes 1.22 to 1.24
- Rancher Kubernetes Engine (RKE):
  - RKE 1.2.16 w/ Rancher 2.5.12 and RKE 1.3.3 w/ 2.6.3
  - RKE 2 (v1.23.6+rke2r2) w/ Rancher 2.6.3
- Red Hat OpenShift Container Platform 4.8, 4.9, or 4.10
- VMware Tanzu Kubernetes Grid 1.4 or 1.5
- VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2 or 1.13

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 the following resources in addition to the environment's resource requirements:

Component	Requirement
Storage backend capacity	At least 500GB available
Worker nodes	At least 3 worker nodes total, with 4 CPU cores and 12GB RAM each
FQDN address	An FQDN address for Astra Control Center

Component	Requirement
Astra Trident	Astra Trident 21.10.1 or newer installed and configured Astra Trident 22.04 or newer for SnapMirror-based application replication

> ⓘ 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.

- **Image registry**: You must have an existing private Docker image registry to which you can push Astra Control Center build images. You need to provide the URL of the image registry where you will upload the images.

- **Astra Trident / ONTAP configuration**: Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP drivers provided by Astra Trident:

    ◦ ontap-nas

    ◦ ontap-nas-flexgroup

    ◦ ontap-san

    ◦ ontap-san-economy

> ⓘ During app cloning in OpenShift environments, Astra Control Center needs to allow OpenShift to mount volumes and change the ownership of files. Because of this, you need to configure an ONTAP volume export policy to allow these operations. You can do so with the following commands:
>
> 1. `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser sys`
>
> 2. `export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -anon 65534`

> ⓘ If you plan to add a second OpenShift operational environment as a managed compute resource, you need to ensure that the Astra Trident Volume Snapshot feature is enabled. To enable and test volume snapshots with Astra Trident, see the official Astra Trident instructions.

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.

- 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.

- As part of Astra Control Center installation, the following resources are created in a pod security policy (PSP) restricted environment:

    ◦ pod security policy

- RBAC Role

- RBAC RoleBinding
  The RBAC Role and RoleBinding resources are created in the `netapp-acc` namespace.

- Be aware of specific requirements for Astra Trident when you deploy Astra Control Center in a TKG or TKGi environment. For more information, see the Astra Trident documentation.

> ⓘ 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.

Google Anthos cluster requirements

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

Supported storage backends

Astra Control Center supports the following storage backends.

- Astra Data Store

- NetApp ONTAP 9.5 or newer AFF and FAS systems

- NetApp Cloud Volumes ONTAP

Application cluster requirements

Astra Control Center has the following requirements for clusters that you plan to manage from Astra Control Center. These requirements also apply if the cluster you plan to manage is the operational environment cluster that hosts Astra Control Center.

- The most recent version of the Kubernetes snapshot-controller component is installed

- An Astra Trident volumesnapshotclass object has been defined by an administrator

- A default Kubernetes storage class exists on the cluster

- At least one storage class is configured to use Astra Trident

> ⓘ Your application cluster should have a `kubeconfig.yaml` file that defines only one *context* element. Visit the Kubernetes documentation for information about creating kubeconfig files.

> ⓘ 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.

Application management requirements

Astra Control has the following application management requirements:

- **Licensing**: To manage applications using Astra Control Center, you need an Astra Control Center license.
- **Namespaces**: Astra Control requires that an app not span more than a single namespace, but a namespace can contain more than one app.
- **StorageClass**: If you install an application with a StorageClass explicitly set and you need to clone the app, the target cluster for the clone operation must have the originally specified StorageClass. Cloning an application with an explicitly set StorageClass to a cluster that does not have the same StorageClass will fail.
- **Kubernetes resources**: Applications that use Kubernetes resources not collected by Astra Control might not have full app data management capabilities. Astra Control collects the following Kubernetes resources:

ClusterRole	ClusterRoleBinding	ConfigMap
CronJob	CustomResourceDefinition	CustomResource
DaemonSet	DeploymentConfig	HorizontalPodAutoscaler
Ingress	MutatingWebhook	NetworkPolicy
PersistentVolumeClaim	Pod	PodDisruptionBudget
PodTemplate	ReplicaSet	Role
RoleBinding	Route	Secret
Service	ServiceAccount	StatefulSet
ValidatingWebhook		

Replication prerequisites

Astra Control application replication requires that the following prerequisites must be met before you begin:

- Astra Control Center must be deployed in a third fault domain or secondary site to achieve seamless disaster recovery.
- The app's host Kubernetes cluster and a destination Kubernetes cluster must be available and connected to two ONTAP clusters, ideally at different failure domains or sites.
- ONTAP clusters and the host SVM must be paired. See Cluster and SVM peering overview.
- The paired remote SVM must be available to Trident on the destination cluster.
- Trident version 22.04 or greater must exist on both the source and destination ONTAP clusters.
- ONTAP SnapMirror asynchronous licenses using the Data Protection bundle must be enabled on both the source and destination ONTAP clusters. See SnapMirror licensing overview in ONTAP.
- The Astra Trident backend configuration file must contain the following line:

```
"replicationPolicy": "MirrorAllSnapshots"
```

See Configure backends for more information.

- When you add an ONTAP storage backend to Astra Control Center, the user whose credentials you use to enable communication between Astra Control Center and the storage backend must have the user login access methods `http` and `ontapi` enabled within ONTAP System Manager on both ONTAP clusters. See Manage User Accounts for more information.

- Both source and destination Kubernetes clusters and ONTAP clusters must be managed by Astra Control.

> (i) You can simultaneously replicate a different app (running on the other cluster or site) in the opposite direction. For example, Apps A, B, C can be replicated from Datacenter 1 to Datacenter 2; and Apps X, Y, Z can be replicated from Datacenter 2 to Datacenter 1.

Learn how to replicate apps to a remote system using SnapMirror technology.

Supported application installation methods

Astra Control supports the following application installation methods:

- **Manifest file**: Astra Control supports apps installed from a manifest file using kubectl. For example:

```
kubectl apply -f myapp.yaml
```

- **Helm 3**: If you use Helm to install apps, Astra Control requires Helm version 3. Managing and cloning apps installed with Helm 3 (or upgraded from Helm 2 to Helm 3) is fully supported. Managing apps installed with Helm 2 is not supported.
- **Operator-deployed apps**: Astra Control supports apps installed with namespace-scoped operators. The following are some apps that have been validated for this installation model:
  - Apache K8ssandra
  - Jenkins CI
  - Percona XtraDB Cluster

> (i) An operator and the app it installs must use the same namespace; you might need to modify the deployment .yaml file for the operator to ensure this is the case.

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 receiving monitoring and metrics data from NetApp Cloud Insights, or sending support bundles to the NetApp Support Site.

License

Astra Control Center requires an Astra Control Center license for full functionality. Obtain an evaluation license or full license from NetApp. Without a license, you can't do any of the following:

- Define apps
- Create snapshots or clones of existing apps
- Configure data protection policies

If you want to try Astra Control Center, you can use a 90-day evaluation license.

To learn more about how licenses work, see Licensing.

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.

> ⓘ If you are hosting Astra Control Center on a Tanzu Kubernetes Grid cluster, 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.

For more information, see [Set up ingress for load balancing](#).

Networking requirements

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 both ways between the cluster hosting Astra Control Center and each managed cluster
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	Hosted Cloud Insights service (https://cloudinsights.netapp.com)	443	HTTPS	Cloud Insights communication

Source	Destination	Port	Protocol	Purpose
Astra Control Center	Amazon S3 storage bucket provider (https://my-bucket.s3.us-west-2.amazonaws.com/)	443	HTTPS	Amazon S3 storage communication
Astra Control Center	NetApp AutoSupport (https://support.netapp.com)	443	HTTPS	NetApp AutoSupport communication

Supported web browsers

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

What's next

View the quick start overview.

Quick start for Astra Control Center

This page provides a high-level 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.

Try it out! If you want to try Astra Control Center, you can use a 90-day evaluation license. See licensing information for details.

1 **Review Kubernetes cluster requirements**

- Astra works with Kubernetes clusters with a Trident-configured ONTAP storage backend or an Astra Data Store storage backend.
- Clusters must be running in a healthy state, with at least three online worker nodes.
- The cluster must be running Kubernetes.

Learn more about the Astra Control Center requirements.

2 **Download and install Astra Control Center**

- Download Astra Control Center from the NetApp Support Site Astra Control Center Downloads page.
- Install Astra Control Center in your local environment.

    Optionally, install Astra Control Center using Red Hat OperatorHub.

Learn more about installing Astra Control Center.

3 **Complete some initial setup tasks**

- Add a license.

- Add a Kubernetes cluster and Astra Control Center discovers details.

- Add an ONTAP or Astra Data Store storage backend.

- Optionally, add an object store bucket that will store your app backups.

Learn more about the initial setup process.

4 **Use Astra Control Center**

After you finish setting up Astra Control Center, here's what you might do next:

- Manage an app. Learn more about how to manage apps.

- Optionally, connect to NetApp Cloud Insights to display metrics on the health of your system, capacity, and throughput inside the Astra Control Center UI. Learn more about connecting to Cloud Insights.

5 **Continue from this Quick Start**

Install Astra Control Center.

Find more information

- Use the Astra Control API

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

Install Astra Control Center using the standard process

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

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

What you'll need

- Before you begin installation, prepare your environment for Astra Control Center deployment.

- If you have configured or want to configure pod security policies in your environment, familiarize yourself with pod security policies and how they affect Astra Control Center installation. See Understand pod security policy restrictions.

- Ensure all cluster operators are in a healthy state and available.

```
kubectl get clusteroperators
```

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

```
kubectl get apiservices
```

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

About this task

The Astra Control Center installation process does the following:

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

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

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

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

Steps

To install Astra Control Center, do the following steps:

- Download and unpack the Astra Control Center bundle
- Install the NetApp Astra kubectl plugin
- Add the images to your local registry
- Set up namespace and secret for registries with auth requirements
- Install the Astra Control Center operator
- Configure Astra Control Center
- Complete Astra Control Center and operator installation

-
-
-

Download and unpack the Astra Control Center bundle

1. Download the Astra Control Center bundle (`astra-control-center-[version].tar.gz`) from the NetApp Support Site.

2. Download the zip of Astra Control Center certificates and keys from the NetApp Support Site.

3. (Optional) Use the following command to verify the signature of the bundle:

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

4. Extract the images:

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

Install the NetApp Astra kubectl plugin

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

What you'll need

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

Steps

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

```
ls kubectl-astra/
```

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

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

Add the images to your local registry

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

Docker

1. Change to the Astra directory:

```
cd acc
```

2. Push the package images in the Astra Control Center image directory to your local registry. Make the following substitutions before running the command:
   - Replace BUNDLE_FILE with the name of the Astra Control bundle file (for example, `acc.manifest.bundle.yaml`).
   - Replace MY_REGISTRY with the URL of the Docker repository.
   - Replace MY_REGISTRY_USER and MY_REGISTRY_PASSWORD with the credentials for the repository.

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

Podman

1. Log in to your registry:

```
podman login [your_registry_path]
```

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

```
# You need to be at the root of the tarball.
# You should see these files to confirm correct location:
#   acc.manifest.bundle.yaml
#   acc/

# Replace <YOUR_REGISTRY> with your own registry (e.g
registry.customer.com or registry.customer.com/testing, etc..)
export REGISTRY=<YOUR_REGISTRY>
export PACKAGENAME=acc
export PACKAGEVERSION=22.08.0-20
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
  # Load to local cache
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')

  # Remove path and keep imageName.
  astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')

  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/netapp/astra/${PACKAGENAME}
/${PACKAGEVERSION}/${astraImageNoPath}

  # Push to the local repo.
  podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

Set up namespace and secret for registries with auth requirements

1. If you use a registry that requires authentication, you need to do the following:

    a. Create the `netapp-acc-operator` namespace:

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

    Response:

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

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

> **(i)** The placeholder `your_registry_path` should match the location of the images that you uploaded earlier (for example, `[Registry_URL]/netapp/astra/astracc/22.08.0-20`).

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

Sample response:

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

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

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

Sample response:

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

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

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

Response

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

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

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

Install the Astra Control Center operator

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

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   > ℹ️ An annotated sample YAML follows these steps.

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

   ```
   imagePullSecrets:
   - name: <name_of_secret_with_creds_to_local_registry>
   ```

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

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [your_registry_path]/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

2. Install the Astra Control Center operator:

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

Sample response:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

Configure Astra Control Center

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

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

   ```
   vim astra_control_center_min.yaml
   ```

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

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

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

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

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

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

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

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

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

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

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

j. Complete one of the following steps:

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

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

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

  This deploys the Astra Control Center `traefik` gateway as a Kubernetes LoadBalancer type service.

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

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

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

Complete Astra Control Center and operator installation

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

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

Sample response:

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

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

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

Sample response:

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

Verify system status

> ℹ️ If you prefer to use OpenShift, you can use comparable oc commands for verification steps.

1. Verify that all system components installed successfully.

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

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

Sample response

```
NAME                                           READY   STATUS    RESTARTS
AGE
acc-helm-repo-6b44d68d94-d8m55                 1/1     Running   0
13m
activity-78f99ddf8-hltct                       1/1     Running   0
10m
api-token-authentication-457nl                 1/1     Running   0
9m28s
api-token-authentication-dgwsz                 1/1     Running   0
9m28s
api-token-authentication-hmqqc                 1/1     Running   0
9m28s
asup-75fd554dc6-m6qzh                          1/1     Running   0
9m38s
authentication-6779b4c85d-92gds                1/1     Running   0
8m11s
bucketservice-7cc767f8f8-lqwr8                 1/1     Running   0
9m31s
certificates-549fd5d6cb-5kmd6                  1/1     Running   0
9m56s
certificates-549fd5d6cb-bkjh9                  1/1     Running   0
9m56s
cloud-extension-7bcb7948b-hn8h2                1/1     Running   0
10m
cloud-insights-service-56ccf86647-fgg69        1/1     Running   0
9m46s
composite-compute-677685b9bb-7vgsf             1/1     Running   0
10m
composite-volume-657d6c5585-dnq79              1/1     Running   0
9m49s
credentials-755fd867c8-vrlmt                   1/1     Running   0
11m
entitlement-86495cdf5b-nwhh2                   1/1     Running   2
10m
features-5684fb8b56-8d6s8                       1/1     Running   0
10m
fluent-bit-ds-rhx7v                            1/1     Running   0
7m48s
fluent-bit-ds-rjms4                            1/1     Running   0
7m48s
fluent-bit-ds-zf5ph                            1/1     Running   0
7m48s
graphql-server-66d895f544-w6hjd                1/1     Running   0
3m29s
```

```
identity-744df448d5-rlcmm                    1/1    Running   0
10m
influxdb2-0                                  1/1    Running   0
13m
keycloak-operator-75c965cc54-z7csw           1/1    Running   0
8m16s
krakend-798d6df96f-9z2sk                     1/1    Running   0
3m26s
license-5fb7d75765-f8mjg                     1/1    Running   0
9m50s
login-ui-7d5b7df85d-l2s7s                    1/1    Running   0
3m20s
loki-0                                       1/1    Running   0
13m
metrics-facade-599b9d7fcc-gtmgl              1/1    Running   0
9m40s
monitoring-operator-67cc74f844-cdplp         2/2    Running   0
8m11s
nats-0                                       1/1    Running   0
13m
nats-1                                       1/1    Running   0
13m
nats-2                                       1/1    Running   0
12m
nautilus-769f5b74cd-k5jxm                    1/1    Running   0
9m42s
nautilus-769f5b74cd-kd9gd                    1/1    Running   0
8m59s
openapi-84f6ccd8ff-76kvp                     1/1    Running   0
9m34s
packages-6f59fc67dc-4g2f5                    1/1    Running   0
9m52s
polaris-consul-consul-server-0               1/1    Running   0
13m
polaris-consul-consul-server-1               1/1    Running   0
13m
polaris-consul-consul-server-2               1/1    Running   0
13m
polaris-keycloak-0                           1/1    Running   0
8m7s
polaris-keycloak-1                           1/1    Running   0
5m49s
polaris-keycloak-2                           1/1    Running   0
5m15s
polaris-keycloak-db-0                        1/1    Running   0
8m6s
```

```
polaris-keycloak-db-1                         1/1    Running  0
5m49s
polaris-keycloak-db-2                         1/1    Running  0
4m57s
polaris-mongodb-0                             2/2    Running  0
13m
polaris-mongodb-1                             2/2    Running  0
12m
polaris-mongodb-2                             2/2    Running  0
12m
polaris-ui-565f56bf7b-zwr8b                   1/1    Running  0
3m19s
polaris-vault-0                               1/1    Running  0
13m
polaris-vault-1                               1/1    Running  0
13m
polaris-vault-2                               1/1    Running  0
13m
public-metrics-6d86d66444-2wbzl               1/1    Running  0
9m30s
storage-backend-metrics-77c5d98dcd-dbhg5 1/1    Running  0
9m44s
storage-provider-78c885f57c-6zcv4             1/1    Running  0
9m36s
telegraf-ds-2l2m9                             1/1    Running  0
7m48s
telegraf-ds-qfzgh                             1/1    Running  0
7m48s
telegraf-ds-shrms                             1/1    Running  0
7m48s
telegraf-rs-bjpkt                             1/1    Running  0
7m48s
telemetry-service-6684696c64-qzfdf            1/1    Running  0
10m
tenancy-6596b6c54d-vmpsm                      1/1    Running  0
10m
traefik-7489dc59f9-6mnst                      1/1    Running  0
3m19s
traefik-7489dc59f9-xrkgg                      1/1    Running  0
3m4s
trident-svc-6c8dc458f5-jswcl                  1/1    Running  0
10m
vault-controller-6b954f9b76-gz9nm             1/1    Running  0
11m
```

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

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

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

3. When all the pods are running, verify installation success by retrieving the `AstraControlCenter` instance installed by the Astra Control Center Operator.

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

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

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

Sample YAML Details

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

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

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

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

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

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

Set up ingress for load balancing

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

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

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

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

- Istio ingress
- Nginx ingress controller
- OpenShift ingress controller

What you'll need

- The required ingress controller should already be deployed.
- The ingress class corresponding to the ingress controller should already be created.
- You are using Kubernetes versions between and including v1.19 and v1.22.

Steps for Istio ingress

1. Configure Istio ingress.

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

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

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

   Sample command:

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

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

   Sample command:

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

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

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

Output:

```
apiVersion: networking.k8s.io/v1beta1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
            serviceName: traefik
            servicePort: 80
```

For the v1 new schema, follow this sample:

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

Output:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. Deploy Astra Control Center as usual.

6. Check the status of the ingress:

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

Steps for Nginx ingress controller

1. Create a secret of type `kubernetes.io/tls` for a TLS private key and certificate in `netapp-acc` (or custom-named) namespace as described in TLS secrets.

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

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

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
          serviceName: traefik
          servicePort: 80
        pathType: ImplementationSpecific
```

b. For the `v1` new schema, follow this sample:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC addess>
    http:
      paths:
        - path:
          backend:
            service:
                name: traefik
                port:
                    number: 80
          pathType: ImplementationSpecific
```

Steps for OpenShift ingress controller

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

2. Create the OpenShift route:

```
oc create route edge --service=traefik
--port=web -n [netapp-acc or custom namespace]
--insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

Log in to the Astra Control Center UI

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

Steps

1. In a browser, enter the FQDN you used in the `astraAddress` in the `astra_control_center_min.yaml` CR when you installed Astra Control Center.

2. Accept the self-signed certificates when prompted.

> (i) You can create a custom certificate after login.

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

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

4. Select **Login**.

5. Change the password when prompted.

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

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

Troubleshoot the installation

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

Steps

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

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

What's next

Complete the deployment by performing [setup tasks](#).

Understand Astra Control Center cluster CR options

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

Setting	Type	Use	Value Example	Description
astraVersion	string	Required	1.5.2	Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version.

Setting	Type	Use	Value Example	Description
astraAddress	string	Required	astra.example.com	Defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center.
accountName	string	Required	Example	Astra Control Center account name. There can be only one.
email	string	Required	admin@example.com	The username of the administrator to be added as the first user of Astra. This email address will be notified by Astra Control as events warrant.
firstName	string	Required	SRE	The first name of the administrator supporting Astra.
lastName	string	Required	Admin	The last name of the administrator supporting Astra.
storageClass	string	Optional (this is the default value)	ontap-gold	The storage class to be used for PVCs. If not set, the default storage class will be used.
volumeReclaimPolicy	Undefined	Optional	Retain	Reclaim policy to be set for persistent volumes.
astraResourcesScaler	string	Required	Default	Scaling options for AstraControlCenter Resource limits. See setting complexities to understand how this settings affects others settings.

Setting	Type	Use	Value Example	Description
astraKubeConfigSecret	string	Required	acc-kubeconfig-cred	If this value is present and a secret exists, the operator will attempt to add that KubeConfig to become the first managed cluster.
ingressType	string	Optional	Generic (this is the default value)	The type of ingress Astra Control Center should be configured for. Valid values are `Generic` and `AccTraefik`. See [setting complexities](#) to understand how this settings affects others settings.
avpDeploy	Boolean	Optional	true (this is the default value)	Option that allows a user to disable deployment of Astra Plugin for VMware vSphere operator.
imageRegistry	Undefined	Optional		The container image registry that is hosting the Astra application images, Astra Control Center Operator, and Astra Control Center Helm Repository.
imageRegistry.name	string	Required if you are using imageRegistry	example.registry.com/astra	The name of the image registry. Do not prefix with protocol.
imageRegistry.secret	string	Required if you are using imageRegistry	astra-registry-cred	The name of the Kubernetes secret used to authenticate with the image registry.
autoSupport	Undefined	Required		Indicates participation status in NetApp's pro-active support application, NetApp Active IQ. An internet connection is required (port 442) and all support data is anonymized.

Setting	Type	Use	Value Example	Description
autoSupport.enrolled	Boolean	Optional, but either `enrolled` or `url` fields must be selected	false (this value is the default)	Enrolled determines if you want to send anonymous data to NetApp for support purposes. The default election is `false` and indicates no support data will be sent to NetApp.
autoSupport.url	string	Optional, but either `enrolled` or `url` fields must be selected	https://support.netapp.com/asupprod/post/1.0/postAsup	URL determines where the anonymous data will be sent.
crds	Undefined	Undefined		Options for how Astra Control Center should handle CRDs.
crds.externalTraefik	Boolean	Optional	True (this value is the default)	By default, Astra Control Center will install the required Traefik CRDs. CRDs are cluster-wide objects and installing them may have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.

Setting	Type	Use	Value Example	Description
crds.externalCertManager	Boolean	Optional	True (this value is the default)	By default, Astra Control Center will install the required cert-manager CRDs. CRDs are cluster-wide objects and installing them may have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.
crds.shouldUpgrade	Boolean	Optional	Undefined	Determines if CRDs should be upgraded when Astra Control Center is upgraded.
mtls				Options for how Astra Control Center should implement service to service mTLS in the cluster. See setting complexities to understand how this settings affects others settings
mtls.enabled	Boolean	Optional	true (this value is the default)	By default, Astra Control Center uses mTLS for service-to-service communication. This option should be disabled when using a service mesh to encrypt service-to-service communication instead.

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

Configuration combinations and incompatibilities

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

astraResourcesScaler

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

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

ingressType

There are two valid values for ingressType:

- Generic
- AccTraefik

Generic (default)

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

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

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

AccTraefik

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

mtls

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

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

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

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

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

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

Understand pod security policy restrictions

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

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

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

PSPs installed by Astra Control Center

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

PSPs created during installation

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

The new policy and objects have the following attributes:

```
$ kubectl get psp
NAME                            PRIV     CAPS          SELINUX     RUNASUSER
FSGROUP      SUPGROUP     READONLYROOTFS    VOLUMES
avp-psp                         false                  RunAsAny    RunAsAny
RunAsAny     RunAsAny     false             *
netapp-astra-deployment-psp     false                  RunAsAny    RunAsAny
RunAsAny     RunAsAny     false             *

$ kubectl get role
NAME                                      CREATED AT
netapp-astra-deployment-role              2022-06-27T19:34:58Z

$ kubectl get rolebinding
NAME                                      ROLE
AGE
netapp-astra-deployment-rb                Role/netapp-astra-deployment-role
32m
```

PSPs created during backup operations

During backup operations, Astra Control Center creates a dynamic pod security policy, a ClusterRole object,

and a RoleBinding object. These support the backup process, which happens in a separate namespace.

The new policy and objects have the following attributes:

```
$ kubectl get psp
NAME                               PRIV     CAPS
SELINUX     RUNASUSER              FSGROUP     SUPGROUP     READONLYROOTFS
VOLUMES
netapp-astra-backup                false    DAC_READ_SEARCH
RunAsAny    RunAsAny              RunAsAny    RunAsAny     false              *

$ kubectl get role
NAME                 CREATED AT
netapp-astra-backup    2022-07-21T00:00:00Z

$ kubectl get rolebinding
NAME                     ROLE                          AGE
netapp-astra-backup    Role/netapp-astra-backup    62s
```

PSPs created during cluster management

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

The new policy and objects have the following attributes:

```
$ kubectl get psp
NAME                               PRIV     CAPS
SELINUX     RUNASUSER              FSGROUP     SUPGROUP     READONLYROOTFS
VOLUMES
netapp-monitoring-psp-nkmo         true     AUDIT_WRITE,NET_ADMIN,NET_RAW
RunAsAny    RunAsAny              RunAsAny    RunAsAny     false              *

$ kubectl get role
NAME                                     CREATED AT
netapp-monitoring-role-privileged           2022-07-21T00:00:00Z

$ kubectl get rolebinding
NAME                                               ROLE
AGE
netapp-monitoring-role-binding-privileged          Role/netapp-
monitoring-role-privileged              2m5s
```

Enable network communication between namespaces

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

Steps

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

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

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

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

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

   ```
   apiVersion: networking.k8s.io/v1
   kind: NetworkPolicy
   metadata:
     name: acc-avp-network-policy
     namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
   CENTER NAMESPACE NAME
   spec:
     podSelector: {}
     policyTypes:
       - Ingress
     ingress:
       - from:
           - namespaceSelector:
               matchLabels:
                 kubernetes.io/metadata.name: <PLUGIN_NAMESPACE_NAME> #
     REPLACE THIS WITH THE ASTRA PLUGIN FOR VMWARE VSPHERE NAMESPACE NAME
   ```

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

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-operator-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <NETAPP-ACC-OPERATOR> #
REPLACE THIS WITH THE OPERATOR NAMESPACE NAME
```

Remove resource limitations

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

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

Steps

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

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

NAME          AGE    REQUEST                                             LIMIT
pods-high      16s    requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low       15s    requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium    16s    requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. Delete all of the resource quotas by name:

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

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

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

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

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

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

4. Delete the limit ranges by name:

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

Configure an external cert-manager

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

Steps

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

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

Sample response:

```
cert-manager    essential-cert-manager-84446f49d5-sf2zd           1/1
Running    0    6d5h
cert-manager    essential-cert-manager-cainjector-66dc99cc56-9ldmt   1/1
Running    0    6d5h
cert-manager    essential-cert-manager-webhook-56b76db9cc-fjqrq      1/1
Running    0    6d5h
```

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

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

Sample response:

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

3. Create a secret with previously generated files:

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

Sample response:

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

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

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: astra-ca-clusterissuer
  namespace: <cert-manager-namespace>
spec:
  ca:
    secretName: selfsigned-tls
```

```
kubectl apply -f ClusterIssuer.yaml
```

Sample response:

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

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

```
kubectl get ClusterIssuer
```

Sample response:

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

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

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.

What you'll need
- Before you begin installation, prepare your environment for Astra Control Center deployment.
- From your OpenShift cluster, ensure all cluster operators are in a healthy state (`available` is `true`):

  ```
  oc get clusteroperators
  ```

- From your OpenShift cluster, ensure all API services are in a healthy state (`available` is `true`):

  ```
  oc get apiservices
  ```

- Create an FQDN address for Astra Control Center in your data center.
- Obtain the necessary permissions and access to the Red Hat OpenShift Container Platform to perform the installation steps described.
- 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.

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

- Install Astra Control Center

Download and unpack the Astra Control Center bundle

1. Download the Astra Control Center bundle (`astra-control-center-[version].tar.gz`) from the [NetApp Support Site](#).

2. Download the zip of Astra Control Center certificates and keys from the [NetApp Support Site](#).

3. (Optional) Use the following command to verify the signature of the bundle:

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

4. Extract the images:

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

Install the NetApp Astra kubectl plugin

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

What you'll need

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

Steps

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

```
ls kubectl-astra/
```

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

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

Add the images to your local registry
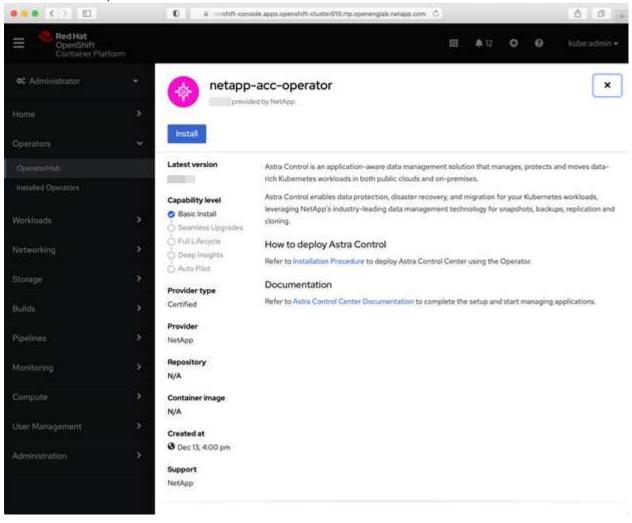
1. Change to the Astra directory:

```
cd acc
```

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

    ◦ Replace BUNDLE_FILE with the name of the Astra Control bundle file (for example, `acc.manifest.bundle.yaml`).

    ◦ Replace MY_REGISTRY with the URL of the Docker repository.

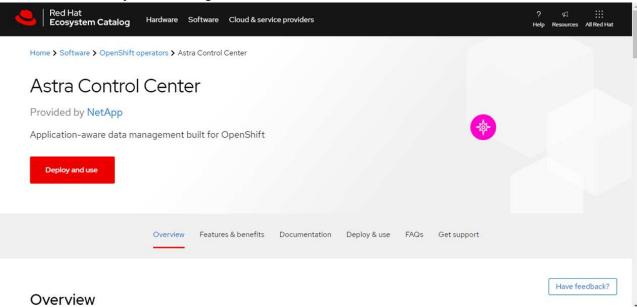   ◦ Replace MY_REGISTRY_USER and MY_REGISTRY_PASSWORD with the credentials for the repository.

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

Find the operator install page

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

    ◦ From Red Hat Openshift web console:

a. Log in to the OpenShift Container Platform UI.

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

c. Select the NetApp Astra Control Center operator.

d. Select **Install**.

◦ From Red Hat Ecosystem Catalog:

a. Select the NetApp Astra Control Center operator.

b. Select **Deploy and Use**.

Install the operator

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

> ⓘ The operator will be available in all cluster namespaces.

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

b. Select a manual or automatic approval strategy.

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

c. Select **Install**.

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

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

Install Astra Control Center

1. From the console within the details view of the Astra Control Center operator, select `Create instance` in

the Provided APIs section.

2. Complete the `Create AstraControlCenter` form field:

   a. Keep or adjust the Astra Control Center name.

   b. (Optional) Enable or disable Auto Support. Retaining Auto Support functionality is recommended.

   c. Enter the Astra Control Center address. Do not enter `http://` or `https://` in the address.

   d. Enter the Astra Control Center version; for example, 21.12.60.

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

   f. Retain the default volume reclaim policy.

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

   h. If you use a registry that requires authentication, enter the secret.

   i. Enter the admin first name.

   j. Configure resources scaling.

   k. Retain the default storage class.

   l. Define CRD handling preferences.

3. Select `Create`.

What's next

Verify the successful installation of Astra Control Center and complete the remaining steps to log in. Additionally, you will complete the deployment by also performing setup tasks.

Install Astra Control Center with a Cloud Volumes ONTAP storage backend

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

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

To install Astra Control Center in Amazon Web Services (AWS) 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 Microsoft Azure

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.

Only self-managed OpenShift Container Platform (OCP) clusters are supported for deploying Astra Control Center.

What you'll need for AWS

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

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

Operational environment requirements for AWS

Astra Control Center requires the following operational environment for AWS:

- Red Hat OpenShift Container Platform 4.8

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

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

Component	Requirement
Backend NetApp Cloud Volumes ONTAP storage capacity	At least 300GB available
Worker nodes (AWS EC2 requirement)	At least 3 worker nodes total, with 4 vCPU cores and 12GB RAM each
Load balancer	Service type "LoadBalancer" available for ingress traffic to be sent to services in the operational environment cluster
FQDN	A method for pointing the FQDN of Astra Control Center to the load balanced IP address
Astra Trident (installed as part of the Kubernetes cluster discovery in NetApp Cloud Manager)	Astra Trident 21.04 or newer installed and configured and NetApp ONTAP version 9.5 or newer as a storage backend
Image registry	You must have an existing private registry, such as AWS Elastic Container Registry, to which you can push Astra Control Center build images. You need to provide the URL of the image registry where you will upload the images. ℹ️ The Astra Control Center hosted cluster and the managed cluster must have access to the same image registry to be able to back up and restore apps using the Restic-based image.

Component	Requirement
Astra Trident / ONTAP configuration	Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP Kubernetes storage classes that are created when you import your Kubernetes cluster into NetApp Cloud Manager. These are provided by Astra Trident: • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

> ℹ️ 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.

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

Overview of deployment for AWS

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

Each of these steps is explained in more detail below.

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

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 Cloud Manager Connector.

See Initial AWS credentials.

Install a RedHat OpenShift cluster on AWS

Install a RedHat OpenShift Container Platform cluster on AWS.

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

Configure AWS

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

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

1. Create an AWS virtual network.

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

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

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

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

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

6. Push the ACC images to your defined registry.

> (i) 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 Cloud Manager

Using Cloud Manager, create a workspace, add a connector to AWS, create a working environment, and import the cluster.

Follow the Cloud Manager documentation to complete the following steps. See the following:

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

Steps

1. Add your credentials to Cloud Manager.

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 Trident version.

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

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

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

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

Install Astra Control Center

Follow the standard Astra Control Center installation instructions.

Deploy Astra Control Center in Microsoft Azure

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

What you'll need for Azure

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

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

Operational environment requirements for Azure

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

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

See Astra Control Center operational environment requirements.

Component	Requirement
Backend NetApp Cloud Volumes ONTAP storage capacity	At least 300GB available
Worker nodes (Azure compute requirement)	At least 3 worker nodes total, with 4 vCPU cores and 12GB RAM each
Load balancer	Service type "LoadBalancer" available for ingress traffic to be sent to services in the operational environment cluster

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

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

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 Cloud Manager.

6. Install and configure Astra Control Center.

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.

> (i) 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 Cloud Manager Connector.

See Azure credentials and permissions.

Configure Azure

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

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

1. Create an Azure virtual network.
2. Review the compute instances. This can be a bare metal server or VMs in Azure.
3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in Azure to meet the Astra requirements. See Astra Control Center requirements.
4. Create at least one Azure Blob container to store your backups.
5. Create a storage account. You will need a storage account to create a container to be used as a bucket in Astra Control Center.
6. Create a secret, which is required for bucket access.
7. Create an Azure Container Registry (ACR) to host all the Astra Control Center images.
8. Set up ACR access for Docker push/pull all the Astra Control Center images.
9. Push the ACC images to this registry by entering the following script:

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

Example:

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

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

10. Set up DNS zones.

Configure NetApp Cloud Manager

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

Follow the Cloud Manager documentation to complete the following steps. See Getting started with Cloud Manager in Azure.
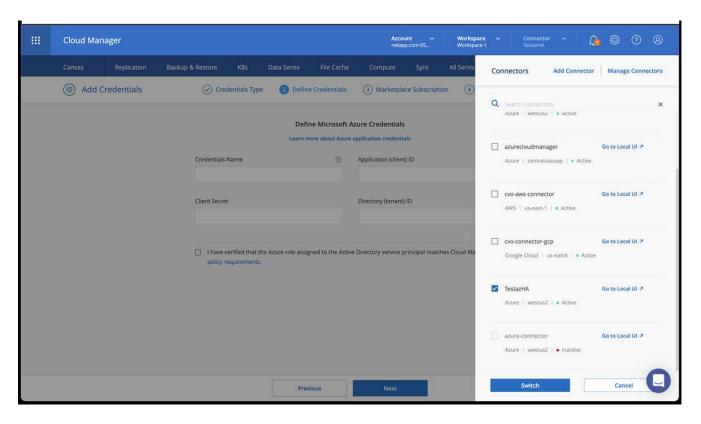
What you'll need
Access to the Azure account with the required IAM permissions and roles

Steps
1. Add your credentials to Cloud Manager.

2. Add a connector for Azure. See Cloud Manager 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 Cloud Manager.

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 Trident version.

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

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

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

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

Install and configure Astra Control Center

Install Astra Control Center with the standard installation instructions.

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

Set up Astra Control Center

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

Tasks

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

Add a license for Astra Control Center

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

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

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

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

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

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

What you'll need

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

Steps

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

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

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

Add cluster

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

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

> (i) When Astra Control manages a cluster, it keeps track of the cluster's default storage class. If you change the storage class using `kubectl` commands, Astra Control reverts the change. To change the default storage class in a cluster managed by Astra Control, use one of the following methods:
>
>   - Use the Astra Control API `PUT /managedClusters` endpoint, and assign a different default storage class with the `DefaultStorageClass` parameter.
>   - Use the Astra Control web UI to assign a different default storage class. See Change the default storage class.
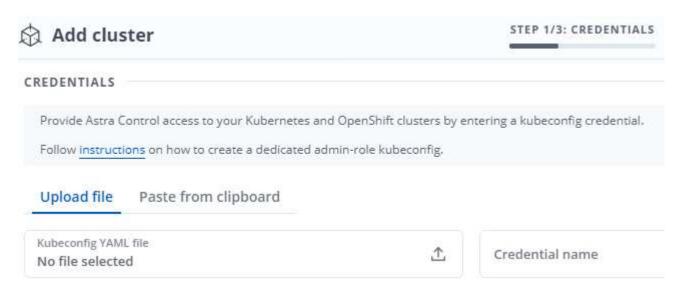
What you'll need

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

Steps

1. From the **Dashboard** in the Astra Control Center UI, select **Add** in the Clusters section.

2. In the **Add Cluster** window that opens, upload a `kubeconfig.yaml` file or paste the contents of a `kubeconfig.yaml` file.

   > (i) The `kubeconfig.yaml` file should include **only the cluster credential for one cluster**.

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

3. Provide a credential name. By default, the credential name is auto-populated as the name of the cluster.

4. Select **Configure storage**.

5. Select the storage class to be used for this Kubernetes cluster, and select **Review**.

   > (i) You should select a Trident storage class backed by ONTAP storage or Astra Data Store.

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

Result

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

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

Add a storage backend

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

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

What you'll need for existing Astra Data Store deployments

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

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

What you'll need for new Astra Data Store deployments

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

Options

- Deploy storage resources
- Use an existing storage backend

Deploy storage resources

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

Steps

1. Navigate from the Dashboard or the Backends menu:

   ◦ From **Dashboard**: From the Resource Summary, select a link from the Storage Backends pane and select **Add** from the Backends section.

   ◦ From **Backends**:

      a. In the left navigation area, select **Backends**.

      b. Select **Add**.

2. Select the **Astra Data Store** deployment option within the **Deploy** tab.

3. Select the Astra Data Store package to deploy:

   a. Enter a name for the Astra Data Store application.

   b. Choose the version of Astra Data Store you want to deploy.

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

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

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

5. If you have not added a Kubernetes cluster to Astra Control management, you need to do so from the **Kubernetes cluster** page. Select an existing cluster from the list or select **add the underlying cluster** to add a cluster to Astra Control management.

6. Select a template size for the Kubernetes cluster that will provide resources for Astra Data Store. You can choose one of the following:

   ◦ If you choose `Recommended Kubernetes worker node requirements`, select a template from large to small based on what your license allows.

   ◦ If you choose `Custom Kubernetes worker node requirements`, select the number of cores and total memory you want for each cluster node. You can also show the eligible number of nodes in the cluster that meet your selection criteria for cores and memory.

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

7. Configure the nodes:

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

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

    b. Configure the capacity (GiB) per node manually or select the maximum node capacity allowed.

    c. Configure a maximum number of nodes allowed in the cluster or allow the maximum number of nodes on the cluster.

8. (Astra Data Store full licenses only) Enter the key of the label you want to use for Protection Domains.

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

9. Configure the management network:

    a. Enter a management IP address for Astra Data Store internal management that is on the same subnet as worker node IP addresses.

    b. Choose to use the same NIC for both management and data networks or configure them separately.

    c. Enter data network IP address pool, subnet mask and gateway for storage access.

10. Review the configuration and select **Deploy** to begin installation.

Result

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

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

Use an existing storage backend

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

Steps

1. Navigate from the Dashboard or the Backends menu:

    ◦ From **Dashboard**: From the Resource Summary, select a link from the Storage Backends pane and select **Add** from the Backends section.

    ◦ From **Backends**:

        a. In the left navigation area, select **Backends**.

b. Select **Manage** on a discovered backend from the managed cluster or select **Add** to manage an additional existing backend.

2. Select the **Use existing** tab.

3. Do one of the following depending on your backend type:

    ◦ **Astra Data Store**:

        a. Select **Astra Data Store**.

        b. Select the managed compute cluster and select **Next**.

        c. Confirm the backend details and select **Add storage backend**.

    ◦ **ONTAP**:

        a. Select **ONTAP** and select **Next**.

        b. Enter the ONTAP cluster management IP address and admin credentials.

> (i) 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, enable the access methods `ontapi` and `http` for the user on both ONTAP clusters. See Manage User Accounts for more information.

        c. Select **Review**.

        d. Confirm the backend details and select **Add storage backend**.

Result

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

> (i) You might need to refresh the page for the backend to appear.

Add a bucket

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

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

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

Use any of the following bucket types:

- NetApp ONTAP S3
- NetApp StorageGRID S3
- Generic S3
- Microsoft Azure

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

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

Steps

1. In the left navigation area, select **Buckets**.

   a. Select **Add**.

   b. Select the bucket type.

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

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

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

   d. Enter the name or IP address of the S3 endpoint.

   e. If you want this bucket to be the default bucket for all backups, check the `Make this bucket the default bucket for this private cloud` option.

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

   f. Continue by adding credential information.

Add S3 access credentials

Add S3 access credentials at any time.

Steps

1. From the Buckets dialog, select either the **Add** or **Use existing** tab.

   a. Enter a name for the credential that distinguishes it from other credentials in Astra Control.

   b. Enter the access ID and secret key by pasting the contents from your clipboard.

Change the default storage class

You can change the default storage class for a cluster.

Steps

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

What's next?

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

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

Find more information

- Use the Astra Control API
- Known issues

Prerequisites for adding a cluster

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

What you'll need before you add a cluster

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

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

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

Run eligibility checks

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

Steps

1. Check the Trident version.

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

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

```
NAME       VERSION
trident    21.04.0
```

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

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

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

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

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

Create an admin-role kubeconfig

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

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

Steps

1. Create a service account as follows:

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

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

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

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

b. Apply the service account:

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

2. (Optional) If your cluster uses a restrictive pod security policy that doesn't allow privileged pod creation or allow processes within the pod containers to run as the root user, create a custom pod security policy for the cluster that enables Astra Control to create and manage pods. For instructions, see Create a custom pod security policy.

3. Grant cluster admin permissions as follows:

    a. Create a `ClusterRoleBinding` file called `astracontrol-clusterrolebinding.yaml`.

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

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

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

b. Apply the cluster role binding:

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

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

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

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

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

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

5. Generate the kubeconfig as follows:

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

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

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

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

   CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

What's next?

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

Find more information

- Trident documentation
- Use the Astra Control API

Add a custom TLS certificate

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

What you'll need

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

Remove the self-signed certificate

Remove the existing self-signed TLS certificate.

1. Using SSH, log in to the Kubernetes cluster that hosts Astra Control Center as an administrative user.

2. Find the TLS secret associated with the current certificate using the following command, replacing `<ACC-deployment-namespace>` with the Astra Control Center deployment namespace:

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

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

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

Add a new certificate

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

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

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

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

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

#spec:
#   selfSigned: {}

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

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

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

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

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

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

5. Create the certificate using the following command:

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

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

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

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

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

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

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

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

8. Using a web browser, browse to the deployment IP address of Astra Control Center.

9. Verify that the certificate details match the details of the certificate you installed.

10. Export the certificate and import the result into the certificate manager in your web browser.

Create a custom pod security policy

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

Steps

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

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

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

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

3. Bind the new role to the service account.

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

Frequently asked questions for Astra Control Center

This FAQ can help if you're just looking for a quick answer to a question.

Overview

The following sections provide answers to some additional questions that you might come across as you use Astra Control Center. For additional clarifications, please reach out to astra.feedback@netapp.com

Access to Astra Control Center

What's the Astra Control URL?

Astra Control Center uses local authentication and a URL specific to each environment.

For the URL, in a browser, enter the Fully Qualified Domain Name (FQDN) you set in the spec.astraAddress field in the astra_control_center_min.yaml custom resource definition (CRD) file when you installed Astra Control Center. The email is the value that you set in the spec.email field in the astra_control_center_min.yaml CRD.

Licensing

I am using the Evaluation license. How to I change to the full license?

You can easily change to a full license by obtaining the NetApp license file (NLF).

Steps

- From the left navigation, select **Account** > **License**.
- Select **Add license**.
- Browse to the license file you downloaded and select **Add**.

I am using the Evaluation license. Can I still manage apps?

Yes, you can test out the managing apps functionality with the Evaluation license.

Registering Kubernetes clusters

I need to add worker nodes to my Kubernetes cluster after adding to Astra Control. What should I do?

New worker nodes can be added to existing pools. These will be automatically discovered by Astra Control. If the new nodes are not visible in Astra Control, check if the new worker nodes are running the supported image type. You can also verify the health of the new worker nodes by using the `kubectl get nodes` command.

How do I properly unmanage a cluster?

1. Unmanage the applications from Astra Control.
2. Unmanage the cluster from Astra Control.

What happens to my applications and data after removing the Kubernetes cluster from Astra Control?

Removing a cluster from Astra Control will not make any changes to the cluster's configuration (applications

and persistent storage). Any Astra Control snapshots or backups taken of applications on that cluster will be unavailable to restore. Persistent storage backups created by Astra Control remain within Astra Control, but they are unavailable for restore.

> ⚠️ Always remove a cluster from Astra Control before you delete it through any other methods. Deleting a cluster using another tool while it's still being managed by Astra Control can cause problems for your Astra Control account.

Is NetApp Trident automatically uninstalled from a cluster when I unmanage it?
When you unmanage a cluster from Astra Control Center, Trident isn't automatically uninstalled from the cluster. To uninstall Trident, you'll need to follow these steps in the Trident documentation.

Managing applications

Can Astra Control deploy an application?

Astra Control doesn't deploy applications. Applications must be deployed outside of Astra Control.

What happens to applications after I stop managing them from Astra Control?

Any existing backups or snapshots will be deleted. Applications and data remain available. Data management operations will not be available for unmanaged applications or any backups or snapshots that belong to it.

Can Astra Control manage an application that is on non-NetApp storage?

No. While Astra Control can discover applications that are using non-NetApp storage, it can't manage an application that's using non-NetApp storage.

Should I manage Astra Control itself?
No, you should not manage Astra Control itself because it is a "system app."

Do unhealthy pods affect app management?
If a managed app has pods in an unhealthy state, Astra Control can't create new backups and clones.

Data management operations

There are snapshots in my account that I didn't create. Where did they come from?

In some situations, Astra Control will automatically create a snapshot as part of a backup, clone or restore process.

My application uses several PVs. Will Astra Control take snapshots and backups of all these PVCs?

Yes. A snapshot operation on an application by Astra Control includes snapshot of all the PVs that are bound to the application's PVCs.

Can I manage snapshots taken by Astra Control directly through a different interface or object storage?

No. Snapshots and backups taken by Astra Control can only be managed with Astra Control.