



Astra Control Center documentation

Astra Control Center

NetApp
June 01, 2022

This PDF was generated from <https://docs.netapp.com/us-en/astra-control-center/index.html> on June 01, 2022. Always check docs.netapp.com for the latest.

Table of Contents

Astra Control Center documentation	1
Release notes	2
What's new in this release of Astra Control Center	2
Known issues	3
Known limitations	5
Concepts	9
Learn about Astra Control	9
Architecture and components	12
Data protection	13
Licensing	14
Validated vs standard apps	15
Storage classes and persistent volume size	16
User roles and namespaces	16
Get started	18
Astra Control Center requirements	18
Quick start for Astra Control Center	23
Installation overview	24
Set up Astra Control Center	60
Frequently asked questions for Astra Control Center	79
Use Astra	81
Manage apps	81
Protect apps	87
View app and cluster health	109
Manage your account	111
Manage buckets	121
Manage the storage backend	124
Monitor and protect infrastructure	128
Unmanage apps and clusters	135
Upgrade Astra Control Center	136
Uninstall Astra Control Center	147
Automate with REST API	151
Automation using the Astra Control REST API	151
Deploy apps	152
Deploy Jenkins from a Helm chart	152
Deploy MariaDB from a Helm chart	153
Deploy MySQL from a Helm chart	154
Deploy Postgres from a Helm chart	156
Knowledge and support	158
Troubleshooting	158
Get help	158
Earlier versions of Astra Control Center documentation	161
Legal notices	162
Copyright	162

Trademarks	162
Patents	162
Privacy policy	162
Open source	162
Astra Control API license	162

Astra Control Center documentation

Release notes

We're pleased to announce the 22.04.0 release of Astra Control Center.

- [What's in this release of Astra Control Center](#)
- [Known issues](#)
- [Known issues with Astra Data Store and this Astra Control Center release](#)
- [Known limitations](#)

Follow us on Twitter [@NetAppDoc](#). Send feedback about documentation by becoming a [GitHub contributor](#) or sending an email to doccomments@netapp.com.

What's new in this release of Astra Control Center

We're pleased to announce the latest 22.04.0 release of Astra Control Center.

26 April 2022 (22.04.0)

New features and support

- [Astra Store deployment from Astra Control Center](#)
- [Namespace role-based access control \(RBAC\)](#)
- [Support for Cloud Volumes ONTAP](#)
- [Generic ingress enablement for Astra Control Center](#)
- [Bucket removal from Astra Control](#)
- [Support for VMware Tanzu Portfolio](#)

Known issues and limitations

- [Known issues for this release](#)
- [Known issues with Astra Data Store and this Astra Control Center release](#)
- [Known limitations for this release](#)

14 December 2021 (21.12)

New features and support

- [Application restore](#)
- [Execution hooks](#)
- [Support for applications deployed with namespace-scoped operators](#)
- [Additional support for upstream Kubernetes and Rancher](#)
- [Astra Data Store preview backend management and monitoring](#)
- [Astra Control Center upgrades](#)
- [Red Hat OperatorHub option for installation](#)

Resolved issues

- [Resolved issues for this release](#)

Known issues and limitations

- [Known issues for this release](#)
- [Known issues with Astra Data Store preview and this Astra Control Center release](#)
- [Known limitations for this release](#)

5 August 2021 (21.08)

Initial release of Astra Control Center.

- [What it is](#)
- [Understand architecture and components](#)
- [What it takes to get started](#)
- [Install and setup](#)
- [Manage and protect apps](#)
- [Manage buckets and storage backends](#)
- [Manage accounts](#)
- [Automate with API](#)

Find more information

- [Known issues for this release](#)
- [Known limitations for this release](#)
- [Astra Data Store documentation](#)
- [Earlier versions of Astra Control Center documentation](#)

Known issues

Known issues identify problems that might prevent you from using this release of the product successfully.

The following known issues affect the current release:

Apps

- [Restore of an app results in PV size larger than original PV](#)
- [App clones fail using a specific version of PostgreSQL](#)
- [App clones fail when using Service Account level OCP Security Context Constraints \(SCC\)](#)
- [App clones fail after an application is deployed with a set storage class](#)

Clusters

- [Managing a cluster with Astra Control Center fails when default kubeconfig file contains more than one context](#)

Other issues

- [App data management operations fail with Internal Service Error \(500\) when Astra Trident is offline](#)
- [Snapshots might fail with snapshot controller version 4.2.0](#)

Restore of an app results in PV size larger than original PV

If you resize a persistent volume after creating a backup and then restore from that backup, the persistent volume size will match the new size of the PV instead of using the size of the backup.

App clones fail using a specific version of PostgreSQL

App clones within the same cluster consistently fail with the Bitnami PostgreSQL 11.5.0 chart. To clone successfully, use an earlier or later version of the chart.

App clones fail when using Service Account level OCP Security Context Constraints (SCC)

An application clone might fail if the original security context constraints are configured at the service account level within the namespace on the OpenShift Container Platform cluster. When the application clone fails, it appears in the Managed Applications area in Astra Control Center with status `Removed`. See the [knowledgebase article](#) for more information.

App clones fail after an application is deployed with a set storage class

After an application is deployed with a storage class explicitly set (for example, `helm install ...-set global.storageClass=netapp-cvs-perf-extreme`), subsequent attempts to clone the application require that the target cluster have the originally specified storage class.

Cloning an application with an explicitly set storage class to a cluster that does not have the same storage class will fail. There are no recovery steps in this scenario.

Managing a cluster with Astra Control Center fails when default kubeconfig file contains more than one context

You cannot use a kubeconfig with more than one cluster and context in it. See the [knowledgebase article](#) for more information.

App data management operations fail with Internal Service Error (500) when Astra Trident is offline

If Astra Trident on an app cluster goes offline (and is brought back online) and 500 internal service errors are encountered when attempting app data management, restart all of the Kubernetes nodes in the app cluster to restore functionality.

Snapshots might fail with snapshot controller version 4.2.0

When you use Kubernetes snapshot-controller (also known as external-snapshotter) version 4.2.0 with Kubernetes 1.20 or 1.21, snapshots can eventually begin to fail. To prevent this, use a different [supported version](#) of external-snapshotter, such as version 4.2.1, with Kubernetes versions 1.20 or 1.21.

1. Run a POST call to add an updated kubeconfig file to the `/credentials` endpoint and retrieve the assigned `id` from the response body.
2. Run a PUT call from the `/clusters` endpoint using the appropriate cluster ID and set the `credentialID` to the `id` value from the previous step.

After you complete these steps, the credential associated with the cluster is updated and the cluster should reconnect and update its state to `available`.

Find more information

- [Known issues with Astra Data Store preview and this Astra Control Center release](#)
- [Known limitations](#)

Known issues with Astra Data Store and this Astra Control Center release

Known issues identify problems that might prevent you from using this release of the product successfully.

See [these known issues](#) that might affect the management of Astra Data Store with the current release of the Astra Control Center.

Find more information

- [Known issues](#)
- [Known limitations](#)

Known limitations

Known limitations identify platforms, devices, or functions that are not supported by this release of the product, or that do not interoperate correctly with it. Review these limitations carefully.

Cluster management limitations

- [The same cluster cannot be managed by two Astra Control Center instances](#)
- [Astra Control Center cannot manage two identically named clusters](#)

Role-based Access Control (RBAC) limitations

- [A user with namespace RBAC constraints can add and unmanage a cluster](#)
- [A member with namespace constraints cannot access the cloned or restored apps until admin adds the namespace to the constraint](#)

App management limitations

- [App backups that are in progress cannot be stopped](#)
- [Clones of apps installed using pass-by-reference operators can fail](#)
- [In-place restore operations of apps that use a certificate manager are not supported](#)
- [OLM-enabled and cluster-scoped operator deployed apps not supported](#)
- [Apps deployed with Helm 2 are not supported](#)

General limitations

- [S3 buckets in Astra Control Center do not report available capacity](#)
- [Astra Control Center does not validate the details you enter for your proxy server](#)
- [Existing connections to a Postgres pod causes failures](#)
- [Backups and snapshots might not be retained during removal of an Astra Control Center instance](#)

The same cluster cannot be managed by two Astra Control Center instances

If you want to manage a cluster on another Astra Control Center instance, you should first [unmanage the cluster](#) from the instance on which it is managed before you manage it on another instance. After you remove

the cluster from management, verify that the cluster is unmanaged by executing this command:

```
oc get pods -n -netapp-monitoring
```

There should be no pods running in that namespace or the namespace should not exist. If either of those are true, the cluster is unmanaged.

Astra Control Center cannot manage two identically named clusters

If you try to add a cluster with the same name of a cluster that already exists, the operation will fail. This issue most often occurs in a standard Kubernetes environment if you have not changed the cluster name default in Kubernetes configuration files.

As a workaround, do the following:

1. Edit your kubeadm-config ConfigMap:

```
kubectl edit configmaps -n kube-system kubeadm-config
```

2. Change the `clusterName` field value from `kubernetes` (the Kubernetes default name) to a unique custom name.
3. Edit `kubeconfig` (`.kube/config`).
4. Update cluster name from `kubernetes` to a unique custom name (`xyz-cluster` is used in the examples below). Make the update in both `clusters` and `contexts` sections as shown in this example:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
    ExAmPLERb2tCcJZ5K3E2Njk4eQotLExAMpLEORCBDRVJUSUZJQ0FURS0txxxxXX==
    server: https://x.x.x.x:6443
    name: xyz-cluster
contexts:
- context:
    cluster: xyz-cluster
    namespace: default
    user: kubernetes-admin
    name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
```

A user with namespace RBAC constraints can add and unmanage a cluster

A user with namespace RBAC constraints should not be allowed to add or unmanage clusters. Due to a current limitation, Astra does not prevent such users from unmanaging clusters.

A member with namespace constraints cannot access the cloned or restored apps until admin adds the namespace to the constraint

Any `member` user with RBAC constraints by namespace name/ID or by namespace labels can clone or restore an app to a new namespace on the same cluster or to any other cluster in their organization's account. However, the same user cannot access the cloned or restored app in the new namespace. After a new namespace is created by a clone or restore operation, the account admin/owner can edit the `member` user account and update role constraints for the affected user to grant access to the new namespace.

App backups that are in progress cannot be stopped

There is no way to stop a running backup. If you need to delete the backup, wait until it has completed and then use the instructions in [Delete backups](#). To delete a failed backup, use the [Astra Control API](#).

Clones of apps installed using pass-by-reference operators can fail

Astra Control supports apps installed with namespace-scoped operators. These operators are generally designed with a "pass-by-value" rather than "pass-by-reference" architecture. The following are some operator apps that follow these patterns:

- [Apache K8ssandra](#)



For K8ssandra, in-place restore operations are supported. A restore operation to a new namespace or cluster requires that the original instance of the application to be taken down. This is to ensure that the peer group information carried over does not lead to cross-instance communication. Cloning of the app is not supported.

- [Jenkins CI](#)
- [Percona XtraDB Cluster](#)

Note that Astra Control might not be able to clone an operator that is designed with a "pass-by-reference" architecture (for example, the CockroachDB operator). During these types of cloning operations, the cloned operator attempts to reference Kubernetes secrets from the source operator despite having its own new secret as part of the cloning process. The clone operation might fail because Astra Control is unaware of the Kubernetes secrets in the source operator.

In-place restore operations of apps that use a certificate manager are not supported

This release of Astra Control Center does not support in-place restore of apps with certificate managers. Restore operations to a different namespace and clone operations are supported.

OLM-enabled and cluster-scoped operator deployed apps not supported

Astra Control Center does not support application management activities with cluster-scoped operators.

Apps deployed with Helm 2 are not supported

If you use Helm to deploy apps, Astra Control Center requires Helm version 3. Managing and cloning apps deployed with Helm 3 (or upgraded from Helm 2 to Helm 3) is fully supported. For more information, see [Astra Control Center requirements](#).

S3 buckets in Astra Control Center do not report available capacity

Before backing up or cloning apps managed by Astra Control Center, check bucket information in the ONTAP or StorageGRID management system.

Astra Control Center does not validate the details you enter for your proxy server

Ensure that you [enter the correct values](#) when establishing a connection.

Existing connections to a Postgres pod causes failures

When you perform operations on Postgres pods, you shouldn't connect directly within the pod to use the `psql` command. Astra Control requires `psql` access to freeze and thaw the databases. If there is a pre-existing connection, the snapshot, backup, or clone will fail.

Backups and snapshots might not be retained during removal of an Astra Control Center instance

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.

Find more information

- [Known issues](#)
- [Known issues with Astra Data Store and this Astra Control Center release](#)

Concepts

Learn about Astra Control

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

Features

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

- Automatically manage persistent storage
- Create application-aware, on-demand snapshots and backups
- Automate policy-driven snapshot and backup operations
- Migrate applications and data from one Kubernetes cluster to another
- Easily clone an application from production to staging
- Visualize application health and protection status
- Use a user interface or an API to implement your backup and migration workflows

Astra Control continually watches your compute for state changes, so it's aware of any new apps that you add along the way.

Deployment models

Astra Control is available in two deployment models:

- **Astra Control Service:** A NetApp-managed service that provides application-aware data management of Kubernetes clusters in Google Kubernetes Engine (GKE) and Azure Kubernetes Service (AKS).
- **Astra Control Center:** Self-managed software that provides application-aware data management of Kubernetes clusters running in your on-premises environment.

	Astra Control Service	Astra Control Center
How is it offered?	As a fully managed cloud service from NetApp	As software that you download, install, and manage
Where is it hosted?	On a public cloud of NetApp's choice	On your provided Kubernetes cluster
How is it updated?	Managed by NetApp	You manage any updates
What are the app data management capabilities?	Same capabilities on both platforms with exceptions to storage backend or to external services	Same capabilities on both platforms with exceptions to storage backend or to external services

	Astra Control Service	Astra Control Center
What is the storage backend support?	NetApp cloud service offerings	<ul style="list-style-type: none"> • NetApp ONTAP AFF and FAS systems • Astra Data Store as storage backend • Cloud Volumes ONTAP storage backend

Supported apps

NetApp has validated some apps to ensure the safety and consistency of the snapshots and backups.

- [Learn the difference between a validated app and a standard app in Astra Control Service.](#)
- [Learn the difference between a validated app and a standard app in Astra Control Center.](#)

No matter which type of app that you use with Astra Control, you should always test the backup and restore workflow yourself to ensure that you can meet your disaster recovery requirements.

How Astra Control Service works

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

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

- You get started with Astra Control Service by setting up your cloud provider and by registering for an Astra account.
 - For GKE clusters, Astra Control Service uses [NetApp Cloud Volumes Service for Google Cloud](#) or Google Persistent Disks as the storage backend for your persistent volumes.
 - For AKS clusters, Astra Control Service uses [Azure NetApp Files](#) or Azure Disk Storage as the storage backend for your persistent volumes.
- You add your first Kubernetes compute to Astra Control Service. Astra Control Service then does the following:
 - Creates an object store in your cloud provider account, which is where backup copies are stored.

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

- Creates a new admin role and Kubernetes service account on the cluster.
 - Uses that new admin role to install [Astra Trident](#) on the cluster and to create one or more storage classes.
 - If you use Azure NetApp Files or NetApp Cloud Volumes Service for Google Cloud as your storage backend, Astra Control Service uses Astra Trident to provision persistent volumes for your apps.
- At this point, you can add apps to your cluster. Persistent volumes will be provisioned on the new default storage class.
- You then use Astra Control Service to manage these apps, and start creating snapshots, backups, and clones.

Astra Control Service continually watches your compute for state changes, so it's aware of any new apps

that you add along the way.

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

How Astra Control Center works

Astra Control Center runs locally in your own private cloud.

Astra Control Center supports OpenShift Kubernetes clusters with:

- Trident storage backends with ONTAP 9.5 and above
- Astra Data Store storage backends

In a cloud connected environment Astra Control Center uses Cloud Insights to provide advanced monitoring and telemetry. In the absence of a Cloud Insights connection, limited (7-days of metrics) monitoring and telemetry is available in Astra Control Center and also exported to Kubernetes native monitoring tools (such as Prometheus and Grafana) through open metrics end points.

Astra Control Center is fully integrated into the AutoSupport and Active IQ ecosystem to provide users and NetApp Support with troubleshooting and usage information.

You can try Astra Control Center out using a 90-day evaluation license. The evaluation version is supported through email and community (Slack channel) options. Additionally, you have access to Knowledgebase articles and documentation from the in-product support dashboard.

To install and use Astra Control Center, you'll need to meet certain [requirements](#).

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

- You install Astra Control Center in your local environment. Learn more about how to [install Astra Control Center](#).
- You complete some setup tasks such as these:
 - Set up licensing.
 - Add your first cluster.
 - Add storage backend that is discovered when you added the cluster.
 - Add an object store bucket that will store your app backups.

Learn more about how to [set up Astra Control Center](#).

Astra Control Center does this:

- Discovers details about the managed Kubernetes clusters.
- Discovers your Astra Trident or Astra Data Store configuration on the clusters that you choose to manage and lets you monitor the storage backends.
- Discovers apps on those clusters and enables you to manage and protect the apps.

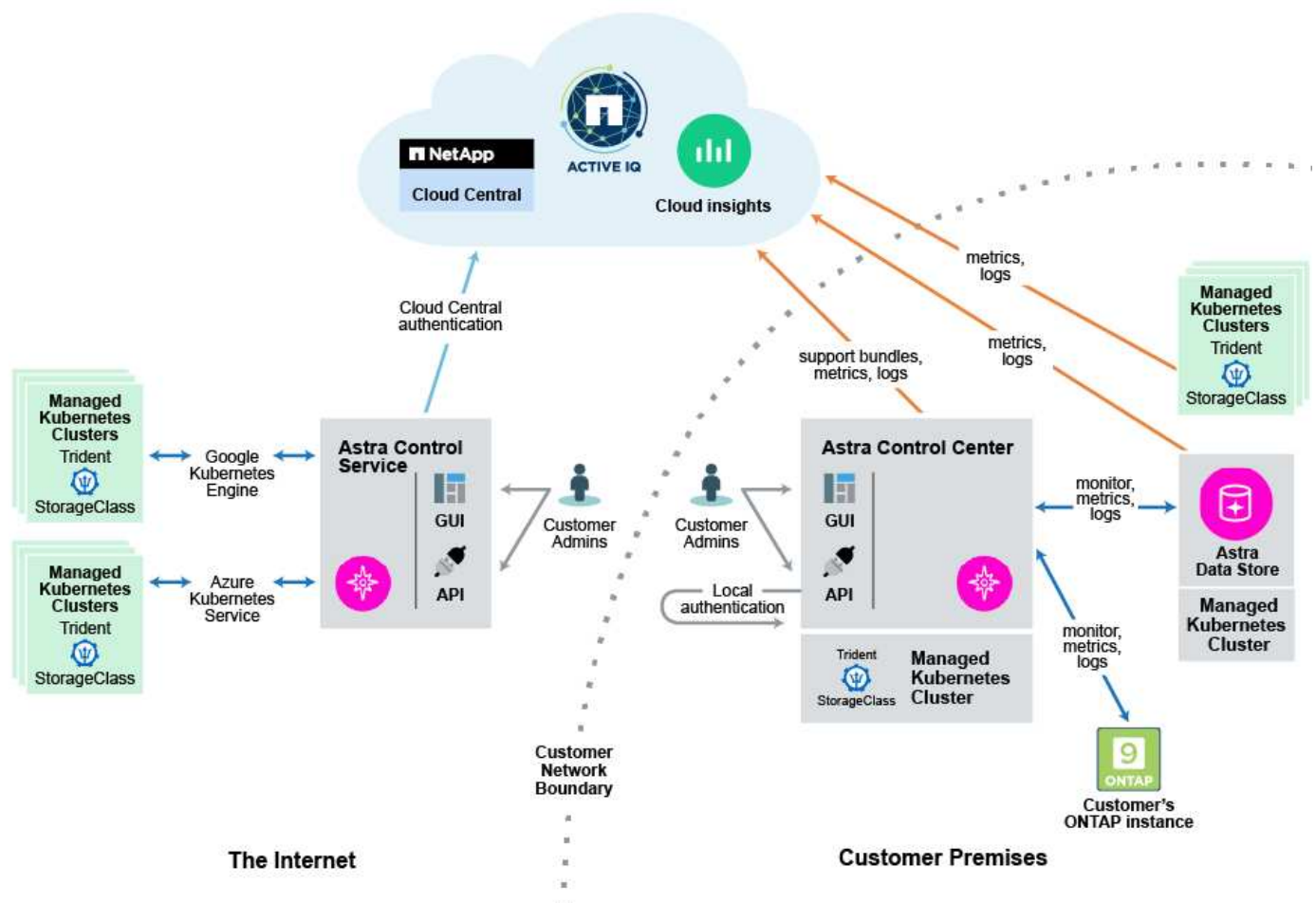
You can add apps to your cluster. Or, if you have some apps already in the cluster being managed, you can use Astra Control Center to discover and manage them. Then, use Astra Control Center to create snapshots, backups, and clones.

For more information

- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Data Store documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [Cloud Insights documentation](#)
- [ONTAP documentation](#)

Architecture and components

Here is an overview of the various components of the Astra Control environment.



Astra Control components

- **Kubernetes clusters:** Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. Astra provides management services for applications hosted in a Kubernetes cluster.
- **Astra Trident:** As a fully supported open source storage provisioner and orchestrator maintained by NetApp, Trident enables you to create storage volumes for containerized applications managed by Docker

and Kubernetes. When deployed with Astra Control Center, Trident includes a configured ONTAP storage backend, and also supports Astra Data Store as a storage backend.

- **Storage backend:**

- Astra Control Service uses [NetApp Cloud Volumes Service for Google Cloud](#) as the storage backend for GKE clusters and [Azure NetApp Files](#) as the storage backend for AKS clusters.
- Astra Control Service also supports Azure Managed Disks and Google Persistent Disk as backend storage options.
- Astra Control Center uses the following storage backends:
 - Astra Data Store storage backend
 - ONTAP AFF and FAS storage backend. As a storage software and hardware platform, ONTAP provides core storage services, support for multiple storage access protocols, and storage management functionality, such as snapshots and mirroring.
 - Cloud Volumes ONTAP storage backend
- **Cloud Insights:** A NetApp cloud infrastructure monitoring tool, Cloud Insights enables you to monitor performance and utilization for your Kubernetes clusters managed by Astra Control Center. Cloud Insights correlates storage usage to workloads. When you enable the Cloud Insights connection in Astra Control Center, telemetry information shows in Astra Control Center UI pages.

Astra Control interfaces

You can complete tasks using different interfaces:

- **Web user interface (UI):** Both Astra Control Service and Astra Control Center use the same web-based UI where you can manage, migrate and protect apps. Use the UI also to manage user accounts and configuration settings.
- **API:** Both Astra Control Service and Astra Control Center use the same Astra Control API. Using the API, you can perform the same tasks that you would using the UI.

Astra Control Center also enables you to manage, migrate, and protect Kubernetes clusters running within VM environments.

For more information

- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [Cloud Insights documentation](#)
- [ONTAP documentation](#)

Data protection

Learn about the available types of data protection in Astra Control Center, and how best to use them to protect your apps.

Snapshots, backups, and protection policies

A *snapshot* is a point-in-time copy of an app that's stored on the same provisioned volume as the app. They are usually fast. You can use local snapshots to restore the application to an earlier point in time. Snapshots are useful for fast clones; snapshots include all of the Kubernetes objects for the app, including configuration files.

A *backup* is stored in the external object store, and can be slower to take compared to local snapshots. You can restore an app backup to the same cluster, or you can migrate an app by restoring its backup to a different cluster. You can also choose a longer retention period for backups. Because they are stored in the external object store, backups generally offer you better protection than snapshots in cases of server failure or data loss.

A *protection policy* is a way to protect an app by automatically creating snapshots, backups, or both according to a schedule that you define for that app. A protection policy also enables you to choose how many snapshots and backups to retain in the schedule. Automating your backups and snapshots with a protection policy is the best way to ensure each app is protected according to the needs of your organization.



You can't be fully protected until you have a recent backup. This is important because backups are stored in an object store away from the persistent volumes. If a failure or accident wipes out the cluster and its associated persistent storage, then you need a backup to recover. A snapshot would not enable you to recover.

Clones

A *clone* is an exact duplicate of an app, its configuration, and its persistent storage. You can manually create a clone on either the same Kubernetes cluster or on another cluster. Cloning an app can be useful if you need to move applications and storage from one Kubernetes cluster to another.

Licensing

Astra Control Center requires a license to be installed for the full App Data Management functionality to be enabled. When you deploy Astra Control Center without a license, a banner is displayed in the web UI, warning that system functionality is limited.

The following operations require a valid license:

- Managing new applications
- Creating snapshots or backups
- Configuring a protection policy to schedule snapshots or backups
- Restoring from a snapshot or backup
- Cloning from a snapshot or current state



You can add a cluster, add a bucket, and manage an Astra Data Store storage backend without a license. However, you need a valid Astra Control Center license to manage apps using Astra Data Store as a storage backend.

How license consumption is calculated

When you add a new cluster to Astra Control Center, it doesn't count toward consumed licenses until at least one application running on the cluster is managed by Astra Control Center. You can also add an Astra Data Store storage backend to Astra Control Center without affecting license consumption. This enables you to manage an Astra Data Store backend from an unlicensed Astra Control Center system.

When you start managing an app on a cluster, the cluster's CPU units are included in the Astra Control Center license consumption calculation.

Find more information

- [Update an existing license](#)

Validated vs standard apps

There are two types of applications you can bring to Astra Control: validated and standard. Learn the difference between these two categories and the potential impacts on your projects and strategy.



It's tempting to think of these two categories as "supported" and "unsupported." But as you will see, there is no such thing as an "unsupported" app in Astra Control. You can add any app to Astra Control, although validated apps have more infrastructure built around their Astra Control workflows compared to standard apps.

Validated apps

Validated apps for Astra Control include the following:

- MySQL 8.0.25
- MariaDB 10.5.9
- PostgreSQL 11.12
- Jenkins 2.277.4 LTS and 2.289.1 LTS

The list of validated apps represents applications that Astra Control recognizes. The Astra Control team has analyzed and confirmed these apps to be fully tested to restore. Astra Control executes custom workflows to help ensure application-level consistency of snapshots and backups.

If an app is validated, the Astra Control team has identified and implemented steps that can be taken to quiesce the app before taking a snapshot in order to obtain an application-consistent snapshot. For example, when Astra Control takes a backup of a PostgreSQL database, it first quiesces the database. After the backup is complete, Astra Control restores the database to normal operation.

No matter which type of app you use with Astra Control, always test the backup and restore workflow yourself to ensure that you can meet your disaster recovery requirements.

Standard apps

Other apps, including custom programs, are considered standard apps. You can add and manage standard apps through Astra Control. You can also create basic, crash-consistent snapshots and backups of a standard app. However, these have not been fully tested to restore the app to its original state.



Astra Control itself is not a standard app; it is a "system app." Astra Control itself isn't shown by default for management. You should not try to manage Astra Control itself.

Storage classes and persistent volume size

Astra Control Center supports ONTAP or Astra Data Store as the storage backend.

Overview

Astra Control Center supports the following:

- **Trident storage classes backed by Astra Data Store storage:** If you installed one or more Astra Data Store clusters manually, Astra Control Center offers the ability to import these and retrieve their topology (nodes, disks) as well as various statuses.

Astra Control Center displays the underlying Kubernetes cluster from the Astra Data Store configuration, the cloud that the Kubernetes cluster belongs to, any persistent volumes provisioned by Astra Data Store, the name of the corresponding internal volume, the application using the persistent volume, and the cluster containing the app.

- **Trident storage classes backed by ONTAP storage:** If you are using an ONTAP backend, Astra Control Center offers the ability to import the ONTAP backend to report various monitoring information.



Trident storage classes should be preconfigured outside of Astra Control Center.

Storage classes

When you add a cluster to Astra Control Center, you're prompted to select one previously configured storage class on that cluster as the default storage class. This storage class will be used when no storage class is specified in a persistent volume claim (PVC). The default storage class can be changed at any time within Astra Control Center and any storage class can be used at any time by specifying the name of the storage class within the PVC or Helm chart. Ensure that you have only a single default storage class defined for your Kubernetes cluster.

When you use Astra Control Center integrated with an Astra Data Store storage backend, after the installation, no storage classes are defined. You will need to create the Trident default storage class and apply it to the storage backend. See [Astra Data Store getting started](#) to create a default Astra Data Store storage class.

For more information

- [Astra Trident documentation](#)

User roles and namespaces

Learn about user roles and namespaces in Astra Control, and how you can use them to control access to resources in your organization.

User roles

You can use roles to control the access users have to resources or capabilities of Astra Control. The following are the user roles in Astra Control:

- A **Viewer** can view resources.
- A **Member** has Viewer role permissions and can manage apps and clusters, unmanage apps, and delete snapshots and backups.
- An **Admin** has Member role permissions and can add and remove any other users except the Owner.
- An **Owner** has Admin role permissions and can add and remove any user accounts.

You can add constraints to a Member or Viewer user to restrict the user to one or more [Namespaces](#).

Namespaces

A namespace is a scope you can assign to specific resources within a cluster that is managed by Astra Control. Astra Control discovers a cluster's namespaces when you add the cluster to Astra Control. Once discovered, the namespaces are available to assign as constraints to users. Only members that have access to that namespace are able to use that resource. You can use namespaces to control access to resources using a paradigm that makes sense for your organization; for example, by physical regions or divisions within a company. When you add constraints to a user, you can configure that user to have access to all namespaces or only a specific set of namespaces. You can also assign namespace constraints using namespace labels.

Find more information

[Manage roles](#)

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

Astra Control Center requires one of the following types of operational environments:

- Kubernetes 1.20 to 1.23
- Rancher 2.5.8, 2.5.9, or 2.6 with RKE1
- Red Hat OpenShift Container Platform 4.6.8, 4.7, 4.8, or 4.9
- VMware Tanzu Kubernetes Grid 1.4
- VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2

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
ONTAP backend storage capacity	At least 300GB 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
Astra Trident	<ul style="list-style-type: none">• Astra Trident 21.04 or newer installed and configured• Astra Trident 21.10.1 or newer installed and configured if Astra Data Store will be used as a storage backend



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.
- You must create a security policy that allows Astra Control Center to create pods within the cluster. You can do this using the following commands:

```
kubectl config use-context <context-of-workload-cluster>
kubectl create clusterrolebinding default-tkg-admin-privileged-binding
--clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

- 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](#).

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

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](#)



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 will be unable to:

- Define custom 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.

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
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 also 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.](#)
- Ensure all cluster operators are in a healthy state and available.

OpenShift example:

```
oc get clusteroperators
```

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

OpenShift example:

```
oc get apiservices
```

- The Astra FQDN you plan to use needs to be 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.

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 of `ACC-<UUID_of_installation>` for this instance of Astra Control Center. This user is assigned the Owner role in the system and is needed for first time login to the UI.
- Helps you determine that all Astra Control Center pods are running.
- Installs the Astra UI.



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



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`

Steps

To install Astra Control Center, do the following steps:

- [Download the Astra Control Center bundle](#)
- [Unpack the bundle and change directory](#)
- [Add the images to your local registry](#)
- [Set up namespace and secret for registries with auth requirements](#)
- [Install the Astra Control Center operator](#)
- [Configure Astra Control Center](#)
- [Complete Astra Control Center and operator installation](#)
- [Verify system status](#)
- [Set up ingress for load balancing](#)
- [Log in to the Astra Control Center UI](#)

Download 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 astra-control-center[version].pub  
-signature <astra-control-center[version].sig astra-control-  
center[version].tar.gz
```

Unpack the bundle and change directory

1. Extract the images:

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

2. Change to the Astra directory.

```
cd astra-control-center-[version]
```

Add the images to your local registry

1. Add the files in the Astra Control Center image directory to your local registry.



See sample scripts for the automatic loading of images below.

- a. Log in to your registry:

Docker:

```
docker login [your_registry_path]
```

Podman:

```
podman login [your_registry_path]
```

- b. Use the appropriate script to load the images, tag the images, and push the images to your local registry:

Docker:

```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded image
    # trimming the 'Loaded images: '
    astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //' )
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    docker tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    docker push ${REGISTRY}/${astraImage}
done
```

Podman:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded image
    trimming the 'Loaded images: '
    astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //'')
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    podman tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    podman push ${REGISTRY}/${astraImage}
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:

```
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
```

- 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:



If additional customizations are required for your environment, you can use `astra_control_center.yaml` as an alternative CR. `astra_control_center_min.yaml` is the default CR and is suitable for most installations.

```
vim astra_control_center_min.yaml
```



Properties configured by the CR cannot be changed after initial Astra Control Center deployment.



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



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

1. Verify that all system components installed successfully.

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

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

Sample response:

NAME	READY	STATUS	RESTARTS
AGE			
acc-helm-repo-5f75c5f564-bzqmt 11m	1/1	Running	0
activity-6b8f7cccb9-mlrn4 9m2s	1/1	Running	0
api-token-authentication-6hznt 8m50s	1/1	Running	0
api-token-authentication-qpfgb 8m50s	1/1	Running	0
api-token-authentication-sqnb7 8m50s	1/1	Running	0
asup-5578bbdd57-dxkbp 9m3s	1/1	Running	0
authentication-56bff4f95d-mspmj 7m31s	1/1	Running	0
bucket-service-6f7968b95d-9rrrl 8m36s	1/1	Running	0
cert-manager-5f6cf4bc4b-82khn 6m19s	1/1	Running	0
cert-manager-cainjector-76cf976458-sdrbc 6m19s	1/1	Running	0
cert-manager-webhook-5b7896bfd8-2n45j 6m19s	1/1	Running	0
cloud-extension-749d9f684c-8bdhq 9m6s	1/1	Running	0
cloud-insights-service-7d58687d9-h5tzw 8m56s	1/1	Running	2
composite-compute-968c79cb5-nv7l4 9m11s	1/1	Running	0
composite-volume-7687569985-jg9gg 8m33s	1/1	Running	0
credentials-5c9b75f4d6-nx9cz	1/1	Running	0

8m42s			
entitlement-6c96fd8b78-zt7f8	1/1	Running	0
8m28s			
features-5f7bfc9f68-gsjnl	1/1	Running	0
8m57s			
fluent-bit-ds-h88p7	1/1	Running	0
7m22s			
fluent-bit-ds-krhnj	1/1	Running	0
7m23s			
fluent-bit-ds-l5bjj	1/1	Running	0
7m22s			
fluent-bit-ds-lrclb	1/1	Running	0
7m23s			
fluent-bit-ds-s5t4n	1/1	Running	0
7m23s			
fluent-bit-ds-zpr6v	1/1	Running	0
7m22s			
graphql-server-5f5976f4bd-vbb4z	1/1	Running	0
7m13s			
identity-56f78b8f9f-8h9p9	1/1	Running	0
8m29s			
influxdb2-0	1/1	Running	0
11m			
krakend-6f8d995b4d-5khkl	1/1	Running	0
7m7s			
license-5b5db87c97-jmxzc	1/1	Running	0
9m			
login-ui-57b57c74b8-6xtv7	1/1	Running	0
7m10s			
loki-0	1/1	Running	0
11m			
monitoring-operator-9dbc9c76d-8znck	2/2	Running	0
7m33s			
nats-0	1/1	Running	0
11m			
nats-1	1/1	Running	0
10m			
nats-2	1/1	Running	0
10m			
nautilus-6b9d88bc86-h8kfb	1/1	Running	0
8m6s			
nautilus-6b9d88bc86-vn68r	1/1	Running	0
8m35s			
openapi-b87d77dd8-5dz9h	1/1	Running	0
9m7s			
polaris-consul-consul-5ljfb	1/1	Running	0

11m			
polaris-consul-consul-s5d5z	1/1	Running	0
11m			
polaris-consul-consul-server-0	1/1	Running	0
11m			
polaris-consul-consul-server-1	1/1	Running	0
11m			
polaris-consul-consul-server-2	1/1	Running	0
11m			
polaris-consul-consul-twmpq	1/1	Running	0
11m			
polaris-mongodb-0	2/2	Running	0
11m			
polaris-mongodb-1	2/2	Running	0
10m			
polaris-mongodb-2	2/2	Running	0
10m			
polaris-ui-84dc87847f-zrg8w	1/1	Running	0
7m12s			
polaris-vault-0	1/1	Running	0
11m			
polaris-vault-1	1/1	Running	0
11m			
polaris-vault-2	1/1	Running	0
11m			
public-metrics-657698b66f-67pgt	1/1	Running	0
8m47s			
storage-backend-metrics-6848b9fd87-w7x8r	1/1	Running	0
8m39s			
storage-provider-5ff5868cd5-r9hj7	1/1	Running	0
8m45s			
telegraf-ds-dw4hg	1/1	Running	0
7m23s			
telegraf-ds-k92gn	1/1	Running	0
7m23s			
telegraf-ds-mmxml	1/1	Running	0
7m23s			
telegraf-ds-nhs8s	1/1	Running	0
7m23s			
telegraf-ds-rj7lw	1/1	Running	0
7m23s			
telegraf-ds-tqrkb	1/1	Running	0
7m23s			
telegraf-rs-9mwgj	1/1	Running	0
7m23s			
telemetry-service-56c49d689b-ffrzz	1/1	Running	0

8m42s	tenancy-767c77fb9d-g9ctv	1/1	Running	0
8m52s	traefik-5857d87f85-7pmx8	1/1	Running	0
6m49s	traefik-5857d87f85-cpxgv	1/1	Running	0
5m34s	traefik-5857d87f85-lvmlb	1/1	Running	0
4m33s	traefik-5857d87f85-t2x1k	1/1	Running	0
4m33s	traefik-5857d87f85-v9wpf	1/1	Running	0
7m3s	trident-svc-595f84dd78-zb816	1/1	Running	0
8m54s	vault-controller-86c94fbf4f-krttq	1/1	Running	0
9m24s				

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

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



`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

```
name: astra
  namespace: netapp-acc
  resourceVersion: "104424560"
  selfLink: /apis/astra.netapp.io/v1/namespaces/netapp-acc/astracontrolcenters/astra
  uid: 9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f
spec:
  accountName: Example
  astraAddress: astra.example.com
  astraVersion: 21.12.60
  autoSupport:
    enrolled: true
    url: https://support.netapp.com/asupprod/post/1.0/postAsup
  crds: {}
  email: admin@example.com
  firstName: SRE
  imageRegistry:
    name: registry_name/astra
    secret: astra-registry-cred
  lastName: Admin
status:
  accConditionHistory:
    items:
      - astraVersion: 21.12.60
        condition:
          lastTransitionTime: "2021-11-23T02:23:59Z"
          message: Deploying is currently in progress.
          reason: InProgress
          status: "False"
          type: Ready
        generation: 2
    observedSpec:
      accountName: Example
      astraAddress: astra.example.com
      astraVersion: 21.12.60
      autoSupport:
        enrolled: true
        url: https://support.netapp.com/asupprod/post/1.0/postAsup
      crds: {}
      email: admin@example.com
      firstName: SRE
      imageRegistry:
        name: registry_name/astra
        secret: astra-registry-cred
```



```

    lastName: Admin
    timestamp: "2021-11-23T02:23:59Z"
- astraVersion: 21.12.60
  condition:
    lastTransitionTime: "2021-11-23T02:23:59Z"
    message: Deploying is currently in progress.
    reason: InProgress
    status: "True"
    type: Deploying
  generation: 2
  observedSpec:
    accountName: Example
    astraAddress: astra.example.com
    astraVersion: 21.12.60
    autoSupport:
      enrolled: true
      url: https://support.netapp.com/asupprod/post/1.0/postAsup
    crds: {}
    email: admin@example.com
    firstName: SRE
    imageRegistry:
      name: registry_name/astra
      secret: astra-registry-cred
    lastName: Admin
    timestamp: "2021-11-23T02:23:59Z"
- astraVersion: 21.12.60
  condition:
    lastTransitionTime: "2021-11-23T02:29:41Z"
    message: Post Install was successful
    observedGeneration: 2
    reason: Complete
    status: "True"
    type: PostInstallComplete
  generation: 2
  observedSpec:
    accountName: Example
    astraAddress: astra.example.com
    astraVersion: 21.12.60
    autoSupport:
      enrolled: true
      url: https://support.netapp.com/asupprod/post/1.0/postAsup
    crds: {}
    email: admin@example.com
    firstName: SRE
    imageRegistry:
      name: registry_name/astra

```

```

    secret: astra-registry-cred
    lastName: Admin
    timestamp: "2021-11-23T02:29:41Z"
- astraVersion: 21.12.60
  condition:
    lastTransitionTime: "2021-11-23T02:29:41Z"
    message: Deploying succeeded.
    reason: Complete
    status: "False"
    type: Deploying
  generation: 2
  observedGeneration: 2
  observedSpec:
    accountName: Example
    astraAddress: astra.example.com
    astraVersion: 21.12.60
    autoSupport:
      enrolled: true
      url: https://support.netapp.com/asupprod/post/1.0/postAsup
    crds: {}
    email: admin@example.com
    firstName: SRE
    imageRegistry:
      name: registry_name/astra
      secret: astra-registry-cred
      lastName: Admin
    observedVersion: 21.12.60
    timestamp: "2021-11-23T02:29:41Z"
- astraVersion: 21.12.60
  condition:
    lastTransitionTime: "2021-11-23T02:29:41Z"
    message: Astra is deployed
    reason: Complete
    status: "True"
    type: Deployed
  generation: 2
  observedGeneration: 2
  observedSpec:
    accountName: Example
    astraAddress: astra.example.com
    astraVersion: 21.12.60
    autoSupport:
      enrolled: true
      url: https://support.netapp.com/asupprod/post/1.0/postAsup
    crds: {}
    email: admin@example.com

```

```

    firstName: SRE
    imageRegistry:
      name: registry_name/astra
      secret: astra-registry-cred
    lastName: Admin
  observedVersion: 21.12.60
  timestamp: "2021-11-23T02:29:41Z"
- astraVersion: 21.12.60
  condition:
    lastTransitionTime: "2021-11-23T02:29:41Z"
    message: Astra is deployed
    reason: Complete
    status: "True"
    type: Ready
  generation: 2
  observedGeneration: 2
  observedSpec:
    accountName: Example
    astraAddress: astra.example.com
    astraVersion: 21.12.60
    autoSupport:
      enrolled: true
      url: https://support.netapp.com/asupprod/post/1.0/postAsup
    crds: {}
    email: admin@example.com
    firstName: SRE
    imageRegistry:
      name: registry_name/astra
      secret: astra-registry-cred
    lastName: Admin
    observedVersion: 21.12.60
    timestamp: "2021-11-23T02:29:41Z"
certManager: deploy
cluster:
  type: OCP
  vendorVersion: 4.7.5
  version: v1.20.0+bafe72f
conditions:
- lastTransitionTime: "2021-12-08T16:19:55Z"
  message: Astra is deployed
  reason: Complete
  status: "True"
  type: Ready
- lastTransitionTime: "2021-12-08T16:19:55Z"
  message: Deploying succeeded.
  reason: Complete

```

```

    status: "False"
    type: Deploying
- lastTransitionTime: "2021-12-08T16:19:53Z"
  message: Post Install was successful
  observedGeneration: 2
  reason: Complete
  status: "True"
  type: PostInstallComplete
- lastTransitionTime: "2021-12-08T16:19:55Z"
  message: Astra is deployed
  reason: Complete
  status: "True"
  type: Deployed
deploymentState: Deployed
observedGeneration: 2
observedSpec:
  accountName: Example
  astraAddress: astra.example.com
  astraVersion: 21.12.60
  autoSupport:
    enrolled: true
    url: https://support.netapp.com/asupprod/post/1.0/postAsup
  crds: {}
  email: admin@example.com
  firstName: SRE
  imageRegistry:
    name: registry_name/astra
    secret: astra-registry-cred
  lastName: Admin
  observedVersion: 21.12.60
  postInstall: Complete
  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.



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:

- 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 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 address>
    http:
      paths:
      - path:
        backend:
          service:
            name: traefik
            port:
              number: 80
        pathType: ImplementationSpecific

```

Steps for OpenShift ingress controller

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

```

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

```

Log in to the Astra Control Center UI

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

Steps

1. In a browser, enter the FQDN you used in the `astraAddress` in the `astra_control_center_min.yaml` CR when [you installed Astra Control Center](#).
2. Accept the self-signed certificates when prompted.



You can create a custom certificate after login.

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



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

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



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

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

Troubleshoot the installation

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

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](#).

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
```

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

Steps

- [Download the Astra Control Center bundle](#)
- [Unpack the bundle and change directory](#)
- [Add the images to your local registry](#)
- [Find the operator install page](#)
- [Install the operator](#)
- [Install Astra Control Center](#)

Download 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 [NetApp Support Site](#).
3. (Optional) Use the following command to verify the signature of the bundle:

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

Unpack the bundle and change directory

1. Extract the images:

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

2. Change to the Astra directory.

```
cd astra-control-center-[version]
```

Add the images to your local registry

1. Add the files in the Astra Control Center image directory to your local registry.



See sample scripts for the automatic loading of images below.

a. Log in to your registry:

Docker:

```
docker login [your_registry_path]
```

Podman:

```
podman login [your_registry_path]
```

b. Use the appropriate script to load the images, tag the images, and push the images to your local registry:

Docker:

```
export REGISTRY=[Docker_registry_path]
for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded image
    trimming the 'Loaded images: '
    astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded
image: //'')
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    docker tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    docker push ${REGISTRY}/${astraImage}
done
```

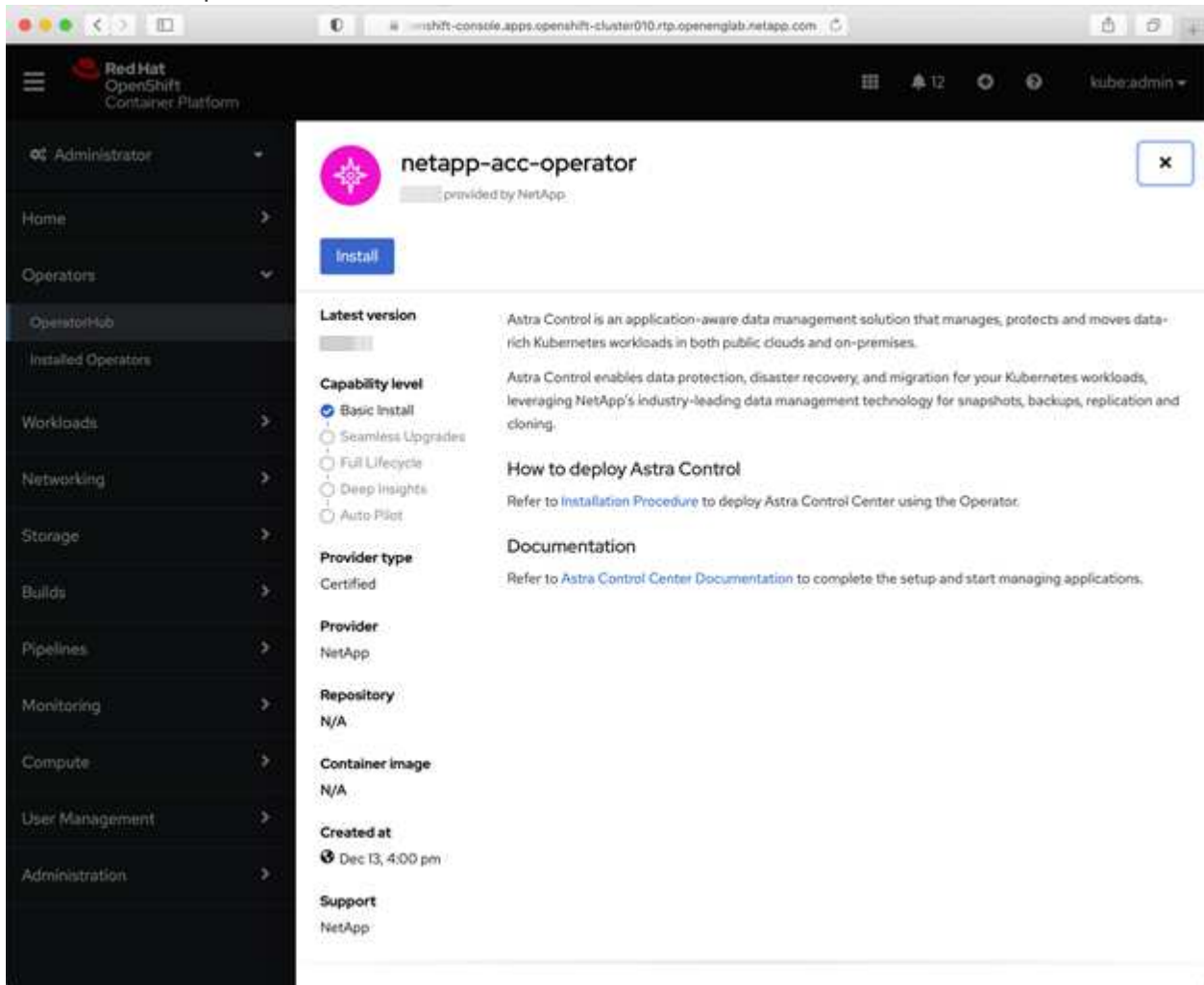
Podman:

```
export REGISTRY=[Registry_path]
for astraImageFile in $(ls images/*.tar) ; do
    # Load to local cache. And store the name of the loaded image
    trimming the 'Loaded images: '
    astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //'')
    astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
    # Tag with local image repo.
    podman tag ${astraImage} ${REGISTRY}/${astraImage}
    # Push to the local repo.
    podman push ${REGISTRY}/${astraImage}
done
```

Find the operator install page

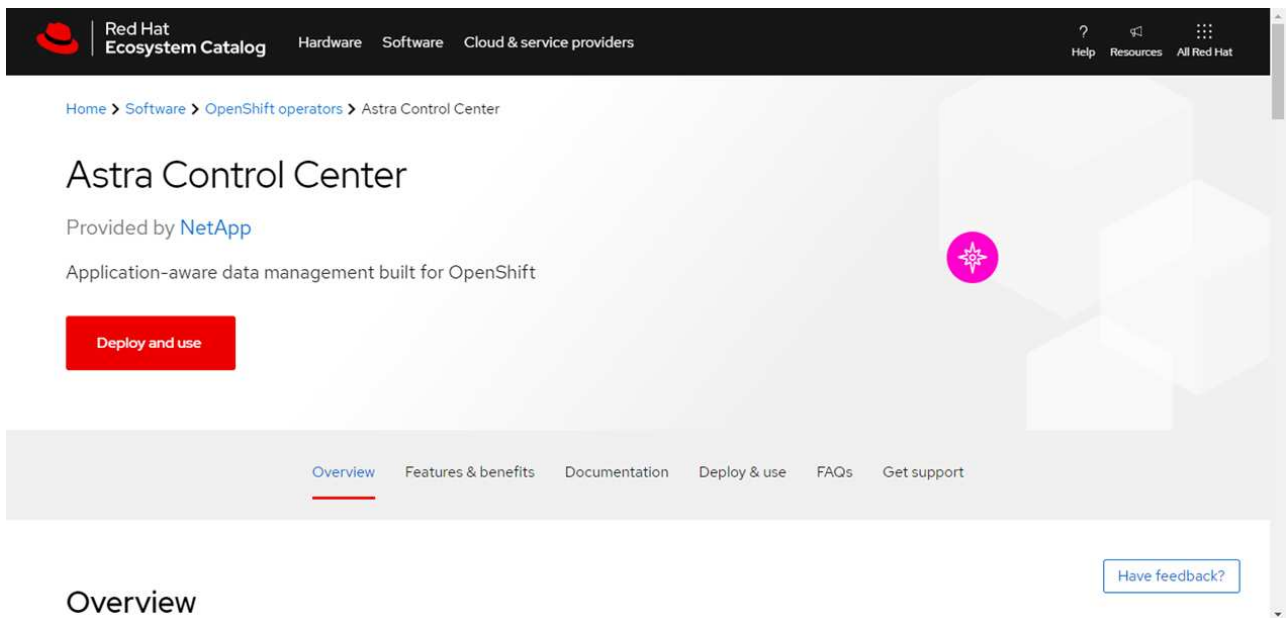
1. Complete one of the following procedures to access the operator install page:

- 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:



Overview

- 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



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	<p>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.</p> <div> <p>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.</p> </div>

Component	Requirement
Astra Trident / ONTAP configuration	<p>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:</p> <ul style="list-style-type: none"> • <code>vsaworkingenvironment-<>-ha-nas</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-ha-san</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-single-nas</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-single-san</code> <code>csi.trident.netapp.io</code>



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



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

Overview of deployment for AWS

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

Each of these steps is explained in more detail below.

1. [Ensure that you have sufficient IAM permissions.](#)
2. [Install a RedHat OpenShift cluster on AWS.](#)
3. [Configure AWS.](#)
4. [Configure NetApp 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.



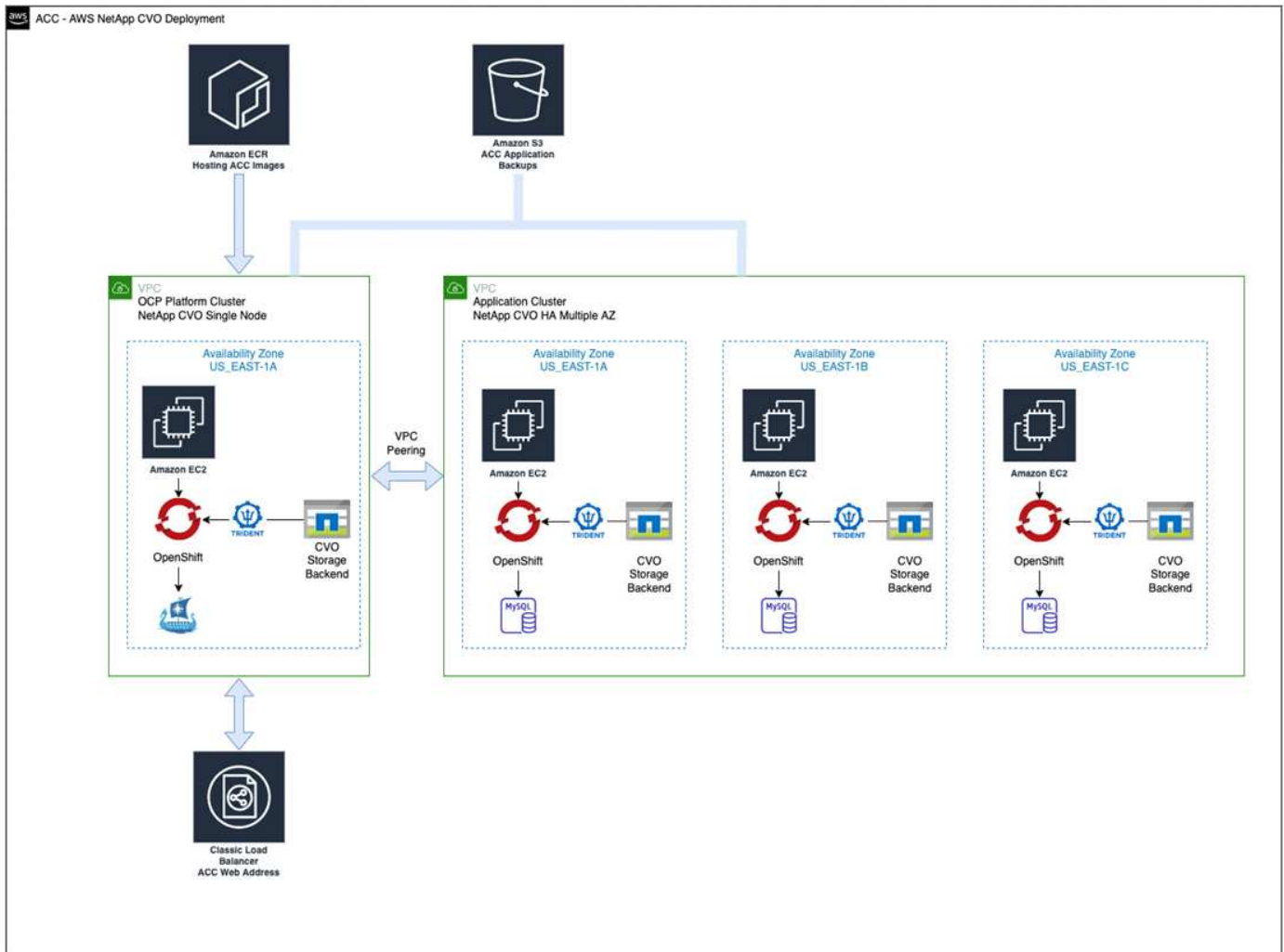
If you do not create the ECR, Astra Control Center cannot access monitoring data from a cluster containing Cloud Volumes ONTAP with an AWS backend. The issue is caused when the cluster you try to discover and manage using Astra Control Center does not have AWS ECR access.

6. Push the ACC images to your defined registry.



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

Here's an example of an AWS deployment:



Configure NetApp 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	<p>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.</p> <div>  <p>You need to enable anonymous access to pull Restic images for backups.</p> </div>
Astra Trident / ONTAP configuration	<p>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:</p> <ul style="list-style-type: none"> • <code>vsaworkingenvironment-<>-ha-nas</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-ha-san</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-single-nas</code> <code>csi.trident.netapp.io</code> • <code>vsaworkingenvironment-<>-single-san</code> <code>csi.trident.netapp.io</code>



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.



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 $AZ_ACR_REGISTRY/$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](#).

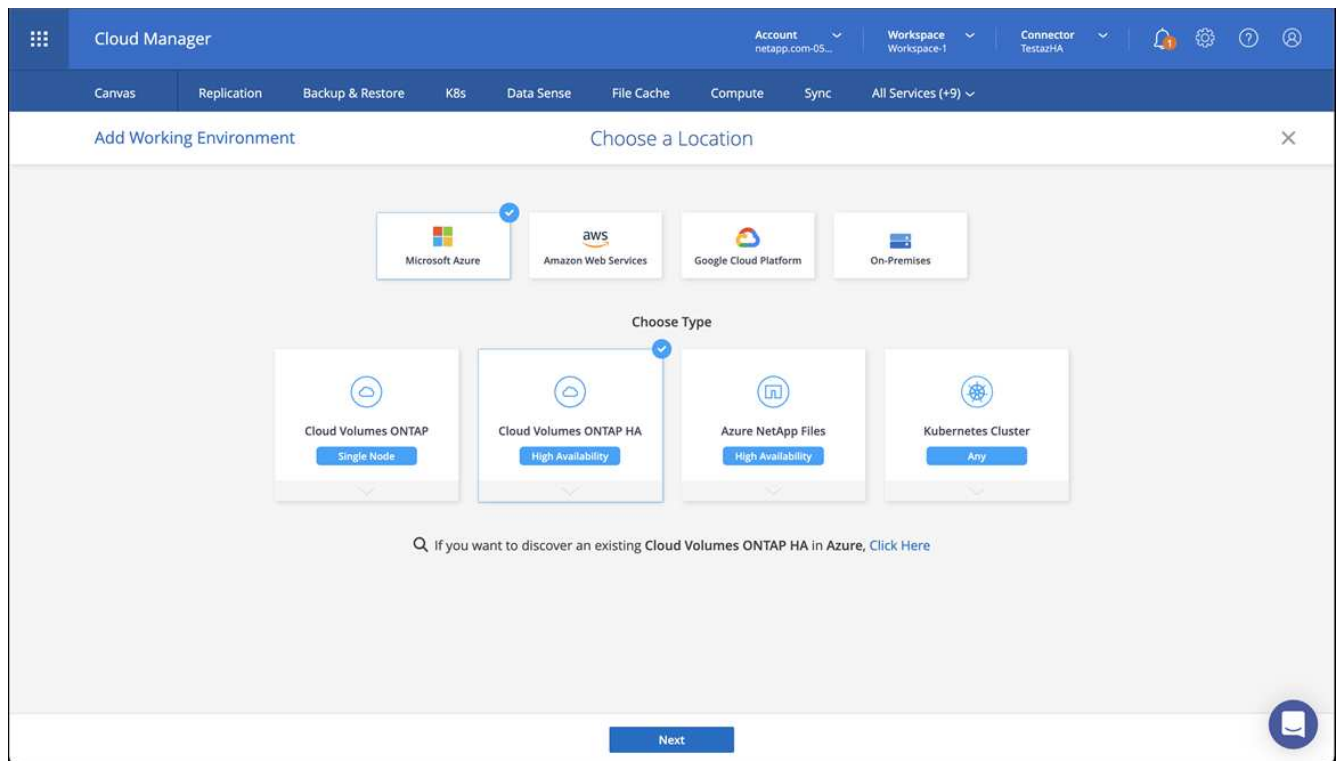
3. Ensure that the connector is running and switch to that connector.



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.

When Astra Control manages a cluster, it keeps track of the cluster's default StorageClass. If you change the StorageClass using `kubectl` commands, Astra Control reverts the change. To change the default StorageClass 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 StorageClass with the `DefaultStorageClass` parameter
- Remove the cluster from Astra Control management and re-add it with a different default StorageClass selected

What you'll need

- Before you add a cluster, review and perform the necessary [prerequisite tasks](#).

Steps

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



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



Add cluster

STEP 1/3: CREDENTIALS

CREDENTIALS

Provide Astra Control access to your Kubernetes and OpenShift clusters by entering a kubeconfig credential.
Follow [instructions](#) on how to create a dedicated admin-role kubeconfig.

Upload file

Paste from clipboard

Kubeconfig YAML file
No file selected



Credential name



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

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



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

CONFIGURE STORAGE

Existing storage classes are discovered and verified as eligible for use with Astra. You can use your existing default, or choose to set a new default at this time.
Applications with persistent volumes on eligible storage classes are validated for use with Astra.

Default	Storage class	Storage provisioner	Reclaim policy	Binding mode	Eligible
<input checked="" type="radio"/>	basic-csi	csi.trident.netapp.io	Delete		
<input type="radio"/>	thin	kubernetes.io/vsphere-volume	Delete		

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

Result

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



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

Add a storage backend

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

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

What you'll need for existing Astra Data Store deployments

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



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

What you'll need for new Astra Data Store deployments

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

Options

- [Deploy storage resources](#)
- [Use an existing storage backend](#)

Deploy storage resources

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

Steps

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



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

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



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

5. If you have not added a Kubernetes cluster to Astra Control management, you need to do so from the **Kubernetes cluster** page. Select an existing cluster from the list or select **add the underlying cluster** to add a cluster to Astra Control management.
6. Select the deployment template size for the Kubernetes cluster that will provide resources for Astra Data Store.



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 template option suggests the number of eligible nodes that satisfy the template pattern for memory and cores and capacity for each node.

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



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

- b. Configure the capacity (GiB) per node manually or select the maximum node capacity allowed.
- c. Configure a maximum number of nodes allowed in the cluster or allow the maximum number of nodes on the cluster.

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



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

9. Configure the management network:

- a. Enter a management IP address for Astra Data Store internal management that is on the same subnet as worker node IP addresses.
- b. Choose to use the same NIC for both management and data networks or configure them separately.
- c. Enter data network IP address pool, subnet mask and gateway for storage access.

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

Result

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



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

Use an existing storage backend

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

Steps

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

- c. Confirm the backend details and select **Add storage backend**.
- **ONTAP:**
 - a. Select **ONTAP**.
 - b. Enter the ONTAP admin credentials and select **Review**.
 - c. Confirm the backend details and select **Add storage backend**.

Result

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



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

Add a bucket

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

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

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

Use any of the following bucket types:

- NetApp ONTAP S3
- NetApp StorageGRID S3
- Generic S3



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

For instructions on how to add buckets using the Astra Control API, see [Astra Automation and API information](#).

Steps

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



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

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



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

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



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

- f. Continue by adding [credential information](#).

Add S3 access credentials

Add S3 access credentials at any time.

Steps

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

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

- One of the following types of clusters:

- Clusters running OpenShift 4.6.8, 4.7, 4.8, or 4.9
- Clusters running Rancher 2.5.8, 2.5.9, or 2.6 with RKE1
- Clusters running Kubernetes 1.20 to 1.23
- Clusters running VMware Tanzu Kubernetes Grid 1.4
- Clusters running VMware Tanzu Kubernetes Grid Integrated Edition 1.12.2

Make sure your clusters have one or more worker nodes with at least 1GB RAM available for running telemetry services.



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

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

Run eligibility checks

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

Steps

1. Check the Trident version.

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

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

NAME	VERSION
trident	21.04.0

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

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



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

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

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

Create an admin-role kubeconfig

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

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

Steps

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

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

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

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

- b. Apply the service account:

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

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

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

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

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

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

- b. Apply the cluster role binding:

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

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

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

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

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

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

5. Generate the kubeconfig as follows:

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

script with the correct value.

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

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

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

CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

# Create token user
```

```

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

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

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

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

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

```

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

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

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

```

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

```

What's next?

Now that you've verified that the prerequisites are met, you're ready to [add a cluster](#).

Find more information

- [Trident documentation](#)
- [Use the Astra Control API](#)

Add a custom TLS certificate

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

What you'll need

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

Remove the self-signed certificate

Remove the existing self-signed TLS certificate.

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

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

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

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

Add a new certificate

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

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

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

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

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

#spec:
#  selfSigned: {}

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

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

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

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

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

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

5. Create the certificate using the following command:

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

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

```

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

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

```

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

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

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

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

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

Create a custom pod security policy

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

Steps

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


```

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

```

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

```

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

```

3. Bind the new role to the service account.

```

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

```

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.

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.

Use Astra

Manage apps

Start managing apps

After you [add a cluster to Astra Control management](#), you can install apps on the cluster (outside of Astra Control), and then go to the Apps page in Astra Control to start managing the apps and their resources.

App management requirements

Astra Control has the following app management requirements:

- **Licensing:** To manage apps 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 app 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:** Apps 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
CustomResourceDefinition	CustomResource	CronJob
DaemonSet	HorizontalPodAutoscaler	Ingress
DeploymentConfig	MutatingWebhook	PersistentVolumeClaim
Pod	PodDisruptionBudget	PodTemplate
NetworkPolicy	ReplicaSet	Role
RoleBinding	Route	Secret
Service	ServiceAccount	StatefulSet
ValidatingWebhook		

Supported app 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) are fully supported. Managing apps installed with Helm 2 is not supported.

- **Operator-deployed apps:** Astra Control supports apps installed with namespace-scoped operators. These operators are generally designed with a "pass-by-value" rather than "pass-by-reference" architecture. The following are some operator apps that follow these patterns:
 - [Apache K8ssandra](#)
 - [Jenkins CI](#)
 - [Percona XtraDB Cluster](#)

Note that Astra Control might not be able to clone an operator that is designed with a "pass-by-reference" architecture (for example, the CockroachDB operator). During these types of cloning operations, the cloned operator attempts to reference Kubernetes secrets from the source operator despite having its own new secret as part of the cloning process. The clone operation might fail because Astra Control is unaware of the Kubernetes secrets in the source operator.



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.

Install apps on your cluster

Now that you've added your cluster to Astra Control, you can install apps or manage existing apps on the cluster. Any app that is scoped to a namespace can be managed. After the pods are online, you can manage the app with Astra Control.

For help with deploying validated apps from Helm charts, refer to the following:

- [Deploy MariaDB from a Helm chart](#)
- [Deploy MySQL from a Helm chart](#)
- [Deploy Postgres from a Helm chart](#)
- [Deploy Jenkins from a Helm chart](#)

Manage apps

Astra Control enables you to manage your apps at the namespace level or by Kubernetes label.



Apps installed with Helm 2 are not supported.

You can perform the following activities to manage apps:

- Manage apps
 - [Manage apps by namespace](#)
 - [Manage apps by Kubernetes label](#)
- [Ignore apps](#)
- [Unmanage apps](#)



Astra Control itself is not a standard app; it is a "system app." You should not try to manage Astra Control itself. Astra Control itself isn't shown by default for management. To see system apps, use the "Show system apps" filter.

For instructions on how to manage apps using the Astra Control API, see the [Astra Automation and API](#)

information.



After a data protection operation (clone, backup, restore) and subsequent persistent volume resize, there is up to a twenty-minute delay before the new volume size is shown in the UI. The data protection operation is successful within minutes, and you can use the management software for the storage backend to confirm the change in volume size.

Manage apps by namespace

The **Discovered** section of the Apps page shows namespaces and any Helm-installed apps or custom-labeled apps in those namespaces. You can choose to manage each app individually or at the namespace level. It all comes down to the level of granularity that you need for data protection operations.

For example, you might want to set a backup policy for "maria" that has a weekly cadence, but you might need to back up "mariadb" (which is in the same namespace) more frequently than that. Based on those needs, you would need to manage the apps separately and not under a single namespace.

While Astra Control enables you to separately manage both levels of the hierarchy (the namespace and the apps in that namespace), the best practice is to choose one or the other. Actions that you take in Astra Control can fail if the actions take place at the same time at both the namespace and app level.

Steps

1. From the left navigation bar, select **Applications**.
2. Select the **Discovered** filter.



3. View the list of discovered namespaces. Expand the namespace to view the apps and associated resources.

Astra Control shows you the Helm apps and custom-labeled apps in the namespace. If Helm labels are available, they're designated with a tag icon.

4. Look at the **Group** column to see which namespace the application is running in (it's designated with the folder icon).
5. Decide whether you want to manage each app individually or at the namespace level.
6. Find the app you want at the desired level in the hierarchy, and select **Manage** from the Options menu in the **Actions** column.
7. If you don't want to manage an app, select **Ignore** from the Options menu in the **Actions** column.

For example, if you want to manage all apps under the "maria" namespace together so that they have the same snapshot and backup policies, you would manage the namespace and ignore the apps in the namespace.

8. To see the list of managed apps, select **Managed** as the display filter.



The app you just added might have a warning icon under the Protected column, indicating that it is not backed up and not scheduled for backups yet.

9. To see details of a particular app, select the app name.

Result

Apps that you chose to manage are now available from the **Managed** tab. Any ignored apps will move to the **Ignored** tab. Ideally, the Discovered tab will show zero apps, so that as new apps are installed, they are easier to find and manage.

Manage apps by Kubernetes label

Astra Control includes an action at the top of the Apps page named **Define custom app**. You can use this action to manage apps that are identified with a Kubernetes label. [Learn more about defining custom apps by Kubernetes label](#).

Steps

1. From the left navigation bar, select **Applications**.
2. Select **Define**.
3. In the **Define custom application** dialog box, provide the required information to manage the app:
 - a. **New App**: Enter the display name of the app.
 - b. **Cluster**: Select the cluster where the app resides.
 - c. **Namespace**: Select the namespace for the app.
 - d. **Label**: Enter a label or select a label from the resources below.
 - e. **Selected Resources**: View and manage the selected Kubernetes resources that you'd like to protect (pods, secrets, persistent volumes, and more).
 - View the available labels by expanding a resource and selecting the number of labels.
 - Select one of the labels.

After you choose a label, it displays in the **Label** field. Astra Control also updates the **Unselected Resources** section to show the resources that don't match the selected label.

- f. **Unselected Resources**: Verify the app resources that you don't want to protect.
4. Select **Define custom application**.

Result

Astra Control enables management of the app. You can now find it in the **Managed** tab.

Ignore apps

If an app has been discovered, it appears in the Discovered list. In this case, you can clean up the Discovered list so that new apps that are newly installed are easier to find. Or, you might have apps that you are managing and later decide you no longer want to manage them. If you don't want to manage these apps, you can indicate that they should be ignored.

Also, you might want to manage apps under one namespace together (Namespace-managed). You can ignore apps that you want to exclude from the namespace.

Steps

1. From the left navigation bar, select **Applications**.
2. Select **Discovered** as the filter.

3. Select the app.
4. From the Options menu in the **Actions** column, select **Ignore**.
5. To unignore, select **Unignore**.

Unmanage apps

When you no longer want to back up, snapshot, or clone an app, you can stop managing it.



If you unmanage an app, any backups or snapshots that were created earlier will be lost.

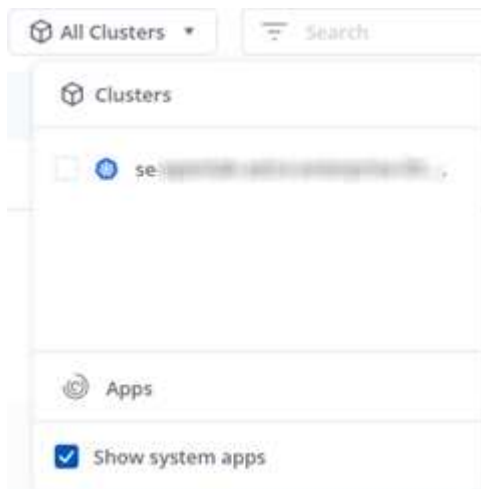
Steps

1. From the left navigation bar, select **Applications**.
2. Select **Managed** as the filter.
3. Select the app.
4. From the Options menu in the **Actions** column, select **Unmanage**.
5. Review the information.
6. Type "unmanage" to confirm.
7. Select **Yes, Unmanage Application**.

What about system apps?

Astra Control also discovers the system apps running on a Kubernetes cluster. We don't show you these system apps by default because it's rare that you'd need to back them up.

You can display system apps from the Applications page by selecting the **Show system apps** check box under the Clusters filter in the toolbar.



Astra Control itself is not a standard app; it is a "system app." You should not try to manage Astra Control itself. Astra Control itself isn't shown by default for management.

Find more information

- [Use the Astra Control API](#)

Define a custom app example

Creating a custom app enables you to group elements of your Kubernetes cluster into a single app. This collection of Kubernetes resources is based on a namespace and a label.

A custom app gives you more granular control over what to include in an Astra Control operation, including:

- Clone
- Snapshot
- Backup
- Protection Policy

In most cases you will want to use Astra Control's features on your entire app. However, you can also create a custom app to use these features by the labels you assign to Kubernetes objects in a namespace.



Custom apps can be created only within a specified namespace on a single cluster. Astra Control does not support the ability for a custom app to span multiple namespaces or clusters.

A label is a key/value pair you can assign to Kubernetes objects for identification. Labels make it easier to sort, organize, and find your Kubernetes objects. To learn more about Kubernetes labels, [see the official Kubernetes documentation](#).



Overlapping policies for the same resource under different names can cause data conflicts. If you create a custom app for a resource, be sure it's not being cloned or backed up under any other policies.

What you'll need

- A cluster added to Astra Control

Steps

1. From the Apps page, select **+ Define**.

The Custom App window shows which resources will be included or excluded from your custom app. This helps you make sure you are choosing the correct criteria for defining your custom app.

2. In the pop-up window, enter the app name, choose the cluster in the **Cluster** drop-down, and choose the app's namespace from the **Namespace** drop-down.
3. From the drop-down **Label** list, select a label for the apps and namespace.
4. After defining the custom app for the one deployment, repeat the process for other deployments, as needed.

When you finish creating the two custom apps, you can treat these resources as any other Astra Control application. They can clone them, create backups and snapshots, and create a custom Protection Policy for each group of resources based on the Kubernetes labels.

Example: Separate Protection Policy for different releases

In this example, the devops team is managing a canary release deployment. Their cluster has three pods running NginX. Two of the pods are dedicated to the stable release. The third pod is for the canary release.

The devops team's Kubernetes admin adds the label `deployment=stable` to the stable release pods. The team adds the label `deployment=canary` to the canary release pod.

The team's stable release includes a requirement for hourly snapshots and daily backups. The canary release is more ephemeral, so they want to create a less aggressive, short-term Protection Policy for anything labeled `deployment=canary`.

In order to avoid possible data conflicts, the admin will create two custom apps: one for the "canary" release, and one for the "stable" release. This keeps the backups, snapshots, and clone operations separate for the two groups of Kubernetes objects.

Protect apps

Protection overview

You can create backups, clones, snapshots, and protection policies for your apps using Astra Control Center. Backing up your apps helps your services and associated data be as available as possible; during a disaster scenario, restoring from backup can ensure full recovery of an app and its associated data with minimal disruption. Backups, clones, and snapshots can help protect against common threats such as ransomware, accidental data loss, and environmental disasters. [Learn about the available types of data protection in Astra Control Center, and when to use them.](#)

App protection workflow

You can use the following example workflow to get started protecting your apps.

[One] Back up all apps

To make sure that your apps are immediately protected, [create a manual backup of all apps](#).

[Two] Configure a protection policy for each app

To automate future backups and snapshots, [configure a protection policy for each app](#). As an example, you can start with weekly backups and daily snapshots, with one month retention for both. Automating backups and snapshots with a protection policy is strongly recommended over manual backups and snapshots.

[Three] Optional: Adjust the protection policies

As apps and their usage patterns change, adjust the protection policies as needed to provide the best protection.

[Four] In case of a disaster, restore your apps

If data loss occurs, you can recover by [restoring the latest backup](#) first for each app. You can then restore the latest snapshot (if available).

Protect apps with snapshots and backups

Protect your apps by taking snapshots and backups using an automated protection policy or on an ad-hoc basis. You can use the Astra UI or [the Astra Control API](#) to protect apps.



If you use Helm to deploy apps, Astra Control Center requires Helm version 3. Managing and cloning apps deployed with Helm 3 (or upgraded from Helm 2 to Helm 3) are fully supported. Apps deployed with Helm 2 are not supported.



When you create a project for hosting an app on an OpenShift cluster, the project (or Kubernetes namespace) is assigned a SecurityContext UID. To enable Astra Control Center to protect your app and move the app to another cluster or project in OpenShift, you need to add policies that enable the app to run as any UID. As an example, the following OpenShift CLI commands grant the appropriate policies to a WordPress app.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

Configure a protection policy

A protection policy protects an app by creating snapshots, backups, or both at a defined schedule. You can choose to create snapshots and backups hourly, daily, weekly, and monthly, and you can specify the number of copies to retain. As an example, a protection policy might create weekly backups and daily snapshots, and retain the backups and snapshots for one month. How often you create snapshots and backups and how long you retain them depends on the needs of your organization.

Steps

1. Select **Applications** and then select the name of an app.
2. Select **Data Protection**.
3. Select **Configure Protection Policy**.
4. Define a protection schedule by choosing the number of snapshots and backups to keep hourly, daily, weekly, and monthly.

You can define the hourly, daily, weekly, and monthly schedules concurrently. A schedule won't turn active until you set a retention level.

The following example sets four protection schedules: hourly, daily, weekly, and monthly for snapshots and backups.

5. Select **Review**.
6. Select **Set Protection Policy**.

Result

Astra Control Center implements the data protection policy by creating and retaining snapshots and backups using the schedule and retention policy that you defined.

Create a snapshot

You can create an on-demand snapshot at any time.

Steps

1. Select **Applications**.

2. From the Options menu in the **Actions** column for the desired app, select **Snapshot**.
3. Customize the name of the snapshot and then select **Review**.
4. Review the snapshot summary and select **Snapshot**.

Result

The snapshot process begins. A snapshot is successful when the status is **Available** in the **Actions** column on the **Data protection > Snapshots** page.

Create a backup

You can also back up an app at any time.



S3 buckets in Astra Control Center do not report available capacity. Before backing up or cloning apps managed by Astra Control Center, check bucket information in the ONTAP or StorageGRID management system.

Steps

1. Select **Applications**.
2. From the Options menu in the **Actions** column for the desired app, select **Backup**.
3. Customize the name of the backup.
4. Choose whether to back up the app from an existing snapshot. If you select this option, you can choose from a list of existing snapshots.
5. Choose a destination for the backup by selecting from the list of storage buckets.
6. Select **Review**.
7. Review the backup summary and select **Backup**.

Result

Astra Control Center creates a backup of the app.



If your network has an outage or is abnormally slow, a backup operation might time out. This causes the backup to fail.



There is no way to stop a running backup. If you need to delete the backup, wait until it has completed and then use the instructions in [Delete backups](#). To delete a failed backup, [use the Astra Control API](#).



After a data protection operation (clone, backup, restore) and subsequent persistent volume resize, there is up to a twenty-minute delay before the new volume size is shown in the UI. The data protection operation is successful within minutes, and you can use the management software for the storage backend to confirm the change in volume size.

View snapshots and backups

You can view the snapshots and backups of an app from the Data Protection tab.

Steps

1. Select **Applications** and then select the name of an app.

2. Select **Data Protection**.

The snapshots display by default.

3. Select **Backups** to see the list of backups.

Delete snapshots

Delete the scheduled or on-demand snapshots that you no longer need.

Steps

1. Select **Applications** and then select the name of an app.
2. Select **Data Protection**.
3. From the Options menu in the **Actions** column for the desired snapshot, select **Delete snapshot**.
4. Type the word "delete" to confirm deletion and then select **Yes, Delete snapshot**.

Result

Astra Control Center deletes the snapshot.

Delete backups

Delete the scheduled or on-demand backups that you no longer need.



There is no way to stop a running backup. If you need to delete the backup, wait until it has completed and then use these instructions. To delete a failed backup, [use the Astra Control API](#).

1. Select **Applications** and then select the name of an app.
2. Select **Data Protection**.
3. Select **Backups**.
4. From the Options menu in the **Actions** column for the desired backup, select **Delete backup**.
5. Type the word "delete" to confirm deletion and then select **Yes, Delete backup**.

Result

Astra Control Center deletes the backup.

Restore apps

Astra Control can restore your application from a snapshot or backup. Restoring from an existing snapshot will be faster when restoring the application to the same cluster. You can use the Astra Control UI or [the Astra Control API](#) to restore apps.

About this task

- It is highly recommended to snapshot or backup your application before proceeding with the restore. This will allow you to clone from the snapshot or backup in the event the restore is unsuccessful.
- If you use Helm to deploy apps, Astra Control Center requires Helm version 3. Managing and cloning apps deployed with Helm 3 (or upgraded from Helm 2 to Helm 3) are fully supported. Apps deployed with Helm 2 are not supported.
- If you restore to a different cluster, ensure that the cluster is using the same persistent volume access

mode (for example, ReadWriteMany). The restore operation will fail if the destination persistent volume access mode is different.

- Any member user with namespace constraints by namespace name/ID or by namespace labels can clone or restore an app to a new namespace on the same cluster or to any other cluster in their organization's account. However, the same user cannot access the cloned or restored app in the new namespace. After a new namespace is created by a clone or restore operation, the account admin/owner can edit the member user account and update role constraints for the affected user to grant access to the new namespace.
- When you create a project for hosting an app on an OpenShift cluster, the project (or Kubernetes namespace) is assigned a SecurityContext UID. To enable Astra Control Center to protect your app and move the app to another cluster or project in OpenShift, you need to add policies that enable the app to run as any UID. As an example, the following OpenShift CLI commands grant the appropriate policies to a WordPress app.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

Steps

1. Select **Applications** and then select the name of an app.
2. Select **Data protection**.
3. If you want to restore from a snapshot, keep the **Snapshots** icon selected. Otherwise, select the **Backups** icon to restore from a backup.
4. From the Options menu in the **Actions** column for the snapshot or backup from which you want to restore, select **Restore application**.
5. **Restore details**: Specify details for the restored app. By default, the current cluster and namespace appear. Leave these values intact to restore an app in-place, which reverts the app to an earlier version of itself. Change these values if you want to restore to a different cluster or namespace.
 - Enter a name and namespace for the app.
 - Choose the destination cluster for the app.
 - Select **Review**.
6. **Restore Summary**: Review details about the restore action, type "restore", and select **Restore**.

Result

Astra Control Center restores the app based on the information that you provided. If you restored the app in-place, the contents of any existing persistent volumes are replaced with the contents of persistent volumes from the restored app.



After a data protection operation (clone, backup, restore) and subsequent persistent volume resize, there is up to a twenty-minute delay before the new volume size is shown in the UI. The data protection operation is successful within minutes, and you can use the management software for the storage backend to confirm the change in volume size.

Clone and migrate apps

Clone an existing app to create a duplicate app on the same Kubernetes cluster or on another cluster. When Astra Control Center clones an app, it creates a clone of your application configuration and persistent storage.

Cloning can help if you need to move applications and storage from one Kubernetes cluster to another. For example, you might want to move workloads through a CI/CD pipeline and across Kubernetes namespaces. You can use the Astra UI or [the Astra Control API](#) to clone and migrate apps.

What you'll need

To clone apps to a different cluster, you need a default bucket. When you add your first bucket, it becomes the default bucket.

About this task

- If you deploy an app with a StorageClass explicitly set and you need to clone the app, the target cluster 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.
- If you clone an operator-deployed instance of Jenkins CI, you need to manually restore the persistent data. This is a limitation of the app's deployment model.
- S3 buckets in Astra Control Center do not report available capacity. Before backing up or cloning apps managed by Astra Control Center, check bucket information in the ONTAP or StorageGRID management system.
- During an app backup or app restore, you can optionally specify a bucket ID. An app clone operation, however, always uses the default bucket that has been defined. There is no option to change buckets for a clone. If you want control over which bucket is used, you can either [change the bucket default](#) or do a [backup](#) followed by a [restore](#) separately.
- Any member user with namespace constraints by namespace name/ID or by namespace labels can clone or restore an app to a new namespace on the same cluster or to any other cluster in their organization's account. However, the same user cannot access the cloned or restored app in the new namespace. After a new namespace is created by a clone or restore operation, the account admin/owner can edit the member user account and update role constraints for the affected user to grant access to the new namespace.

OpenShift considerations

- If you clone an app between clusters, the source and destination clusters must be the same distribution of OpenShift. For example, if you clone an app from an OpenShift 4.7 cluster, use a destination cluster that is also OpenShift 4.7.
- When you create a project for hosting an app on an OpenShift cluster, the project (or Kubernetes namespace) is assigned a SecurityContext UID. To enable Astra Control Center to protect your app and move the app to another cluster or project in OpenShift, you need to add policies that enable the app to run as any UID. As an example, the following OpenShift CLI commands grant the appropriate policies to a WordPress app.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

Steps

1. Select **Applications**.
2. Do one of the following:
 - Select the Options menu in the **Actions** column for the desired app.
 - Select the name of the desired app, and select the status drop-down list at the top right of the page.
3. Select **Clone**.
4. **Clone details**: Specify details for the clone:

- Enter a name.
 - Enter a namespace for the clone.
 - Choose a destination cluster for the clone.
 - Choose whether you want to create the clone from an existing snapshot or backup. If you don't select this option, Astra Control Center creates the clone from the app's current state.
5. **Source:** If you chose to clone from an existing snapshot or backup, choose the snapshot or backup that you'd like to use.
 6. Select **Review**.
 7. **Clone Summary:** Review the details about the clone and select **Clone**.

Result

Astra Control Center clones that app based on the information that you provided. The clone operation is successful when the new app clone is in the `Available` state on the **Applications** page.



After a data protection operation (clone, backup, restore) and subsequent persistent volume resize, there is up to a twenty-minute delay before the new volume size is shown in the UI. The data protection operation is successful within minutes, and you can use the management software for the storage backend to confirm the change in volume size.

Manage app execution hooks

An execution hook is a custom script that you can run before or after a snapshot of a managed app. For example, if you have a database app, you can use execution hooks to pause all database transactions before a snapshot, and resume transactions after the snapshot is complete. This ensures application-consistent snapshots.

Default execution hooks and regular expressions

For some apps, Astra Control comes with default execution hooks, provided by NetApp, that handle freeze and thaw operations before and after snapshots. Astra Control uses regular expressions to match an app's container image to these apps:

- MariaDB
 - Matching regular expression: `\bmariadb\b`
- MySQL
 - Matching regular expression: `\bmysql\b`
- PostgreSQL
 - Matching regular expression: `\bpostgresql\b`

If there is a match, the NetApp-provided default execution hooks for that app appear in the app's list of active execution hooks, and those hooks run automatically when snapshots of that app are taken. If one of your custom apps has a similar image name that happens to match one of the regular expressions (and you don't want to use the default execution hooks), you can either change the image name, or disable the default execution hook for that app and use a custom hook instead.

You cannot delete or modify the default execution hooks.

Important notes about custom execution hooks

Consider the following when planning execution hooks for your apps.

- Astra Control requires execution hooks to be written in the format of executable shell scripts.
- Script size is limited to 128KB.
- Astra Control uses execution hook settings and any matching criteria to determine which hooks are applicable to a snapshot.
- All execution hook failures are soft failures; other hooks and the snapshot are still attempted even if a hook fails. However, when a hook fails, a warning event is recorded in the **Activity** page event log.
- To create, edit, or delete execution hooks, you must be a user with Owner, Admin, or Member permissions.
- If an execution hook takes longer than 25 minutes to run, the hook will fail, creating an event log entry with a return code of "N/A". Any affected snapshot will time out and be marked as failed, with a resulting event log entry noting the timeout.



Since execution hooks often reduce or completely disable the functionality of the application they are running against, you should always try to minimize the time your custom execution hooks take to run.

When a snapshot is run, execution hook events take place in the following order:

1. Any applicable NetApp-provided default pre-snapshot execution hooks are run on the appropriate containers.
2. Any applicable custom pre-snapshot execution hooks are run on the appropriate containers. You can create and run as many custom pre-snapshot hooks as you need, but the order of execution of these hooks before the snapshot is neither guaranteed nor configurable.
3. The snapshot is performed.
4. Any applicable custom post-snapshot execution hooks are run on the appropriate containers. You can create and run as many custom post-snapshot hooks as you need, but the order of execution of these hooks after the snapshot is neither guaranteed nor configurable.
5. Any applicable NetApp-provided default post-snapshot execution hooks are run on the appropriate containers.



You should always test your execution hook scripts before enabling them in a production environment. You can use the 'kubectl exec' command to conveniently test the scripts. After you enable the execution hooks in a production environment, test the resulting snapshots to ensure they are consistent. You can do this by cloning the app to a temporary namespace, restoring the snapshot, and then testing the app.

View existing execution hooks

You can view existing custom or NetApp-provided default execution hooks for an app.

Steps

1. Go to **Applications** and then select the name of a managed app.
2. Select the **Execution hooks** tab.

You can view all enabled or disabled execution hooks in the resulting list. You can see a hook's status, source, and when it runs (pre- or post-snapshot). To view event logs surrounding execution hooks, go to

the **Activity** page in the left-side navigation area.

Create a custom execution hook

You can create a custom execution hook for an app. See [Execution hook examples](#) for hook examples. You need to have Owner, Admin, or Member permissions to create execution hooks.



When you create a custom shell script to use as an execution hook, remember to specify the appropriate shell at the beginning of the file, unless you are running linux commands or providing the full path to an executable.

Steps

1. Select **Applications** and then select the name of a managed app.
2. Select the **Execution hooks** tab.
3. Select **Add a new hook**.
4. In the **Hook Details** area, depending on when the hook should run, choose **Pre-Snapshot** or **Post-Snapshot**.
5. Enter a unique name for the hook.
6. (Optional) Enter any arguments to pass to the hook during execution, pressing the Enter key after each argument you enter to record each one.
7. In the **Container Images** area, if the hook should run against all container images contained within the application, enable the **Apply to all container images** check box. If instead the hook should act only on one or more specified container images, enter the container image names in the **Container image names to match** field.
8. In the **Script** area, do one of the following:
 - Upload a custom script.
 - a. Select the **Upload file** option.
 - b. Browse to a file and upload it.
 - c. Give the script a unique name.
 - d. (Optional) Enter any notes other administrators should know about the script.
 - Paste in a custom script from the clipboard.
 - a. Select the **Paste from clipboard** option.
 - b. Select the text field and paste the script text into the field.
 - c. Give the script a unique name.
 - d. (Optional) Enter any notes other administrators should know about the script.
9. Select **Add hook**.

Disable an execution hook

You can disable an execution hook if you want to temporarily prevent it from running before or after a snapshot of an app. You need to have Owner, Admin, or Member permissions to disable execution hooks.

Steps

1. Select **Applications** and then select the name of a managed app.

2. Select the **Execution hooks** tab.
3. Select the Options menu in the **Actions** column for a hook that you wish to disable.
4. Select **Disable**.

Delete an execution hook

You can remove an execution hook entirely if you no longer need it. You need to have Owner, Admin, or Member permissions to delete execution hooks.

Steps

1. Select **Applications** and then select the name of a managed app.
2. Select the **Execution hooks** tab.
3. Select the Options menu in the **Actions** column for a hook that you wish to delete.
4. Select **Delete**.

Execution hook examples

Use the following examples to get an idea of how to structure your execution hooks. You can use these hooks as templates, or as test scripts.

Simple success example

This is an example of a simple hook that succeeds and writes a message to standard output and standard error.

```
#!/bin/sh

# success_sample.sh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
```

```

#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.sh"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

Simple success example (bash version)

This is an example of a simple hook that succeeds and writes a message to standard output and standard error, written for bash.

```

#!/bin/bash

# success_sample.bash
#
# A simple noop success hook script for testing purposes.
#
# args: None

#
# Writes the given message to standard output
#
# $* - The message to write

```

```

#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.bash"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

Simple success example (zsh version)

This is an example of a simple hook that succeeds and writes a message to standard output and standard error, written for Z shell.

```

#!/bin/zsh

# success_sample.zsh
#
# A simple noop success hook script for testing purposes.

```

```

#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample.zsh"

# exit with 0 to indicate success
info "exit 0"
exit 0

```

Success with arguments example

The following example demonstrates how you can use args in a hook.

```
#!/bin/sh

# success_sample_args.sh
#
# A simple success hook script with args for testing purposes.
#
# args: Up to two optional args that are echoed to stdout
#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running success_sample_args.sh"
```

```
# collect args
arg1=$1
arg2=$2

# output args and arg count to stdout
info "number of args: $#"
```

```
info "arg1 ${arg1}"
info "arg2 ${arg2}"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

Pre-snapshot / post-snapshot hook example

The following example demonstrates how the same script can be used for both a pre-snapshot and a post-snapshot hook.

```
#!/bin/sh

# success_sample_pre_post.sh
#
# A simple success hook script example with an arg for testing purposes
# to demonstrate how the same script can be used for both a prehook and
# posthook
#
# args: [pre|post]

# unique error codes for every error case
ebase=100
eusage=$((ebase+1))
ebadstage=$((ebase+2))
epre=$((ebase+3))
epost=$((ebase+4))

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}
```



```

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# Would run prehook steps here
#
prehook() {
    info "Running noop prehook"
    return 0
}

#
# Would run posthook steps here
#
posthook() {
    info "Running noop posthook"
    return 0
}

#
# main
#

# check arg
stage=$1
if [ -z "${stage}" ]; then
    echo "Usage: $0 <pre|post>"

```

```

        exit ${eusage}
    fi

    if [ "${stage}" != "pre" ] && [ "${stage}" != "post" ]; then
        echo "Invalid arg: ${stage}"
        exit ${ebadstage}
    fi

    # log something to stdout
    info "running success_sample_pre_post.sh"

    if [ "${stage}" = "pre" ]; then
        prehook
        rc=$?
        if [ ${rc} -ne 0 ]; then
            error "Error during prehook"
        fi
    fi

    if [ "${stage}" = "post" ]; then
        posthook
        rc=$?
        if [ ${rc} -ne 0 ]; then
            error "Error during posthook"
        fi
    fi

    exit ${rc}

```

Failure example

The following example demonstrates how you can handle failures in a hook.

```

#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#
#
# Writes the given message to standard output
#

```

```

# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}

```

Verbose failure example

The following example demonstrates how you can handle failures in a hook, with more verbose logging.

```
#!/bin/sh
```

```

# failure_sample_verbose.sh
#
# A simple failure hook script with args for testing purposes.
#
# args: [The number of lines to output to stdout]

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_sample_verbose.sh"

# output arg value to stdout
linecount=$1

```

```

info "line count ${linecount}"

# write out a line to stdout based on line count arg
i=1
while [ "$i" -le ${linecount} ]; do
    info "This is line ${i} from failure_sample_verbose.sh"
    i=$(( i + 1 ))
done

error "exiting with error code 8"
exit 8

```

Failure with an exit code example

The following example demonstrates a hook failing with an exit code.

```

#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#

```

```

# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}

```

Success after failure example

The following example demonstrates a hook failing the first time it is run, but succeeding after the second run.

```

#!/bin/sh

# failure_then_success_sample.sh
#
# A hook script that fails on initial run but succeeds on second run for
# testing purposes.
#
# Helpful for testing retry logic for post hooks.
#
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {

```

```

    echo "$*"
}

#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
# main
#

# log something to stdout
info "running failure_success sample.sh"

if [ -e /tmp/hook-test.junk ] ; then
    info "File does exist. Removing /tmp/hook-test.junk"
    rm /tmp/hook-test.junk
    info "Second run so returning exit code 0"
    exit 0
else
    info "File does not exist. Creating /tmp/hook-test.junk"
    echo "test" > /tmp/hook-test.junk
    error "Failed first run, returning exit code 5"
    exit 5
fi

```

View app and cluster health

View a summary of app and cluster health

Select the **Dashboard** to see a high-level view of your apps, clusters, storage backends, and their health.

These aren't just static numbers or statuses—you can drill down from each. For example, if apps aren't fully protected, you can hover over the icon to identify which apps aren't fully protected, which includes a reason why.

Applications tile

The **Applications** tile helps you identify the following:

- How many apps you're currently managing with Astra.
- Whether those managed apps are healthy.
- Whether the apps are fully protected (they're protected if recent backups are available).
- The number of apps that were discovered, but are not yet managed.

Ideally, this number would be zero because you would either manage or ignore apps after they're discovered. And then you would monitor the number of discovered apps on the Dashboard to identify when developers add new apps to a cluster.

Clusters tile

The **Clusters** tile provides similar details about the health of the clusters that you are managing by using Astra Control Center, and you can drill down to get more details just like you can with an app.

Storage backends tile

The **Storage backends** tile provides information to help you identify the health of storage backends including:

- How many storage backends are managed
- Whether these managed backends are healthy
- Whether the backends are fully protected
- The number of backends that are discovered, but are not yet managed.

View the health and details of clusters

After you add clusters to be managed by Astra Control Center, you can view details about the cluster, such as its location, the worker nodes, persistent volumes, and storage classes.

Steps

1. In the Astra Control Center UI, select **Clusters**.
2. On the **Clusters** page, select the cluster whose details you want to view.



If a cluster is in `removed` state yet cluster and network connectivity appears healthy (external attempts to access the cluster using Kubernetes APIs are successful), the kubeconfig you provided to Astra Control might no longer be valid. This can be due to certificate rotation or expiration on the cluster. To correct this issue, update the credentials associated with the cluster in Astra Control using the [Astra Control API](#).

3. View the information on the **Overview**, **Storage**, and **Activity** tabs to find the information that you're looking for.
 - **Overview**: Details about the worker nodes, including their state.
 - **Storage**: The persistent volumes associated with the compute, including the storage class and state.
 - **Activity**: Shows the activities related to the cluster.



You can also view cluster information starting from the Astra Control Center **Dashboard**. On the **Clusters** tab under **Resource summary**, you can select the managed clusters, which takes you to the **Clusters** page. After you get to the **Clusters** page, follow the steps outlined above.

View the health and details of an app

After you start managing an app, Astra provides details about the app that enables you to identify its status (whether it's healthy), its protection status (whether it's fully protected in case of failure), the pods, persistent storage, and more.

Steps

1. In the Astra Control Center UI, select **Applications** and then select the name of an app.
2. Find the information that you're looking for:

App Status

Provides a status that reflects the app's state in Kubernetes. For example, are pods and persistent volumes online? If an app is unhealthy, you'll need to go and troubleshoot the issue on the cluster by looking at Kubernetes logs. Astra doesn't provide information to help you fix a broken app.

App Protection Status

Provides a status of how well the app is protected:

- **Fully protected**: The app has an active backup schedule and a successful backup that's less than a week old
- **Partially protected**: The app has an active backup schedule, an active snapshot schedule, or a successful backup or snapshot
- **Unprotected**: Apps that are neither fully protected or partially protected.

You can't be fully protected until you have a recent backup. This is important because backups are stored in an object store away from the persistent volumes. If a failure or accident wipes out the cluster and it's persistent storage, then you need a backup to recover. A snapshot wouldn't enable you to recover.

Overview

Information about the state of the pods that are associated with the app.

Data protection

Enables you to configure a data protection policy and to view the existing snapshots and backups.

Storage

Shows you the app-level persistent volumes. The state of a persistent volume is from the perspective of the Kubernetes cluster.

Resources

Enables you to verify which resources are being backed up and managed.

Activity

Shows the activities related to the app.



You can also view app information starting from the Astra Control Center **Dashboard**. On the **Applications** tab under **Resource summary**, you can select the managed apps, which takes you to the **Applications** page. After you get to the **Applications** page, follow the steps outlined above.

Manage your account

Manage users

You can invite, add, remove, and edit users of your Astra Control Center installation using the Astra Control UI. You can use the Astra Control UI or [the Astra Control API](#) to manage users.

Invite users

Account Owners and Admins can invite new users to Astra Control Center.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Users** tab.
3. Select **Invite User**.
4. Enter the user's name and email address.
5. Select a user role with the appropriate system permissions.

Each role provides the following permissions:

- A **Viewer** can view resources.
 - A **Member** has Viewer role permissions and can manage apps and clusters, unmanage apps, and delete snapshots and backups.
 - An **Admin** has Member role permissions and can add and remove any other users except the Owner.
 - An **Owner** has Admin role permissions and can add and remove any user accounts.
6. To add constraints to a user with a Member or Viewer role, enable the **Restrict role to constraints** check box.

For more information on adding constraints, see [Manage roles](#).

7. Select **Invite users**.

The user receives an email informing them that they've been invited to Astra Control Center. The email includes temporary password, which they'll need to change upon first login.

Add users

Account Owners and Admins can add more users to the Astra Control Center installation.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Users** tab.
3. Select **Add User**.
4. Enter the user's name, email address, and a temporary password.

The user will need to change the password upon first login.

5. Select a user role with the appropriate system permissions.

Each role provides the following permissions:

- A **Viewer** can view resources.
 - A **Member** has Viewer role permissions and can manage apps and clusters, unmanage apps, and delete snapshots and backups.
 - An **Admin** has Member role permissions and can add and remove any other users except the Owner.
 - An **Owner** has Admin role permissions and can add and remove any user accounts.
6. To add constraints to a user with a Member or Viewer role, enable the **Restrict role to constraints** check box.

For more information on adding constraints, see [Manage roles](#).

7. Select **Add**.

Manage passwords

You can manage passwords for user accounts in Astra Control Center.

Change your password

You can change the password of your user account at any time.

Steps

1. Select the User icon at the top right of the screen.
2. Select **Profile**.
3. From the Options menu in the **Actions** column, and select **Change Password**.
4. Enter a password that conforms to the password requirements.
5. Enter the password again to confirm.
6. Select **Change password**.

Reset another user's password

If your account has Admin or Owner role permissions, you can reset passwords for other user accounts as well as your own. When you reset a password, you assign a temporary password that the user will have to change upon logging in.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Actions** drop-down list.
3. Select **Reset Password**.
4. Enter a temporary password that conforms to the password requirements.
5. Enter the password again to confirm.



The next time the user logs in, the user will be prompted to change the password.

6. Select **Reset password**.

Change a user's role

Users with the Owner role can change the role of all users, while users with the Admin role can change the role of users who have the Admin, Member, or Viewer role.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Actions** drop-down list.
3. Select **Edit role**.
4. Select a new role.
5. To apply constraints to the role, enable the **Restrict role to constraints** check box and select a constraint from the list.

If there are no constraints, you can add a constraint. For more information, see [Manage roles](#).

6. Select **Confirm**.

Result

Astra Control Center updates the user's permissions based on the new role that you selected.

Remove users

Users with the Owner or Admin role can remove other users from the account at any time.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. In the **Users** tab, select the check box in the row of each user that you want to remove.
3. From the Options menu in the **Actions** column, select **Remove user/s**.
4. When you're prompted, confirm deletion by typing the word "remove" and then select **Yes, Remove User**.

Result

Astra Control Center removes the user from the account.

Manage roles

You can manage roles by adding namespace constraints and restricting user roles to those constraints. This enables you to control access to resources within your organization. You can use the Astra Control UI or [the Astra Control API](#) to manage roles.

Add a namespace constraint to a role

An Admin or Owner user can add namespace constraints.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Users** tab.
3. In the **Actions** column, select the menu button for a user with the Member or Viewer role.
4. Select **Edit role**.
5. Enable the **Restrict role to constraints** check box.

The check box is only available for Member or Viewer roles. You can select a different role from the **Role** drop-down list.

6. Select **Add constraint**.

You can view the list of available constraints by namespace or by namespace label.

7. In the **Constraint type** drop-down list, select either **Kubernetes namespace** or **Kubernetes namespace label** depending on how your namespaces are configured.
8. Select one or more namespaces or labels from the list to compose a constraint that restricts roles to those namespaces.
9. Select **Confirm**.

The **Edit role** page displays the list of constraints you've chosen for this role.

10. Select **Confirm**.

On the **Account** page, you can view the constraints for any Member or Viewer role in the **Role** column.



If you enable constraints for a role and select **Confirm** without adding any constraints, the role is considered to have full restrictions (the role is denied access to any resources that are assigned to namespaces).

Remove a namespace constraint from a role

An Admin or Owner user can remove a namespace constraint from a role.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Users** tab.

3. In the **Actions** column, select the menu button for a user with the Member or Viewer role that has active constraints.
4. Select **Edit role**.

The **Edit role** dialog displays the active constraints for the role.

5. Select the **X** to the right of the constraint you need to remove.
6. Select **Confirm**.

For more information

- [User roles and namespaces](#)

View and manage notifications

Astra notifies you when actions have completed or failed. For example, you'll see a notification if a backup of an app completed successfully.

You can manage these notifications from the top right of the interface:



Steps

1. Select the number of unread notifications in the top right.
2. Review the notifications and then select **Mark as read** or **Show all notifications**.

If you selected **Show all notifications**, the Notifications page loads.

3. On the **Notifications** page, view the notifications, select the ones that you want to mark as read, select **Action** and select **Mark as read**.

Add and remove credentials

Add and remove credentials for local private cloud providers such as ONTAP S3, Kubernetes clusters managed with OpenShift, or unmanaged Kubernetes clusters from your account at any time. Astra Control Center uses these credentials to discover Kubernetes clusters and the apps on the clusters, and to provision resources on your behalf.

Note that all users in Astra Control Center share the same sets of credentials.

Add credentials

You can add credentials to Astra Control Center when you manage clusters. To add credentials by adding a new cluster, see [Add a Kubernetes cluster](#).



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

Remove credentials

Remove credentials from an account at any time. You should only remove credentials after [unmanaging all associated clusters](#).



The first set of credentials that you add to Astra Control Center is always in use because Astra Control Center uses the credentials to authenticate to the backup bucket. It's best not to remove these credentials.

Steps

1. Select **Account**.
2. Select the **Credentials** tab.
3. Select the Options menu in the **State** column for the credentials that you want to remove.
4. Select **Remove**.
5. Type the word "remove" to confirm deletion and then select **Yes, Remove Credential**.

Result

Astra Control Center removes the credentials from the account.

Monitor account activity

You can view details about the activities in your Astra Control account. For example, when new users were invited, when a cluster was added, or when a snapshot was taken. You also have the ability to export your account activity to a CSV file.

View all account activity in Astra Control

1. Select **Activity**.
2. Use the filters to narrow down the list of activities or use the search box to find exactly what you're looking for.
3. Select **Export to CSV** to download your account activity to a CSV file.

View account activity for a specific app

1. Select **Applications** and then select the name of an app.
2. Select **Activity**.

View account activity for clusters

1. Select **Clusters** and then select the name of the cluster.
2. Select **Activity**.

Take action to resolve events that require attention

1. Select **Activity**.
2. Select an event that requires attention.
3. Select the **Take action** drop-down option.

From this list, you can view possible corrective actions that you can take, view documentation related to the issue, and get support to help resolve the issue.

Update an existing license

You can convert an evaluation license to a full license, or you can update an existing evaluation or full license with a new license. If you don't have a full license, work with your NetApp sales contact to obtain a full license and serial number. You can use the Astra UI or [the Astra Control API](#) to update an existing license.

Steps

1. Log in to the [NetApp Support Site](#).
2. Access the Astra Control Center Download page, enter the serial number, and download the full NetApp license file (NLF).
3. Log in to the Astra Control Center UI.
4. From the left navigation, select **Account > License**.
5. In the **Account > License** page, select the status drop-down menu for the existing license and select **Replace**.
6. Browse to the license file that you downloaded.
7. Select **Add**.

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

For more information

- [Astra Control Center licensing](#)

Manage repository connections

You can connect repositories to Astra Control to use as a reference for software package installation images and artifacts. When you import software packages, Astra Control references installation images in the image repository and binaries and other artifacts in the artifact repository.

What you'll need

- Kubernetes cluster with Astra Control Center installed
- A running Docker repository that you can access
- A running artifact repository (such as Artifactory) that you can access

Connect a Docker image repository

You can connect a Docker image repository to hold package installation images, such as those for Astra Data Store. When you install packages, Astra Control imports the package image files from the image repository.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Connections** tab.
3. In the **Docker Image Repository** section, select the menu at the top right.
4. Select **Connect**.
5. Add the URL and port for the repository.
6. If authentication is required, enable the **Use authentication** check box and enter the credentials for the repository.

7. Select **Connect**.

Result

The repository is connected. In the **Docker Image Repository** section, the repository should show a connected status.

Disconnect a Docker image repository

You can remove the connection to a Docker image repository if it is no longer needed.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Connections** tab.
3. In the **Docker Image Repository** section, select the menu at the top right.
4. Select **Disconnect**.
5. Select **Yes, disconnect Docker image repository**.

Result

The repository is disconnected. In the **Docker Image Repository** section, the repository should show a disconnected status.

Connect an artifact repository

You can connect an artifact repository to host artifacts such as software package binaries. When you install packages, Astra Control imports the artifacts for the software packages from the image repository.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Connections** tab.
3. In the **Artifact Repository** section, select the menu at the top right.
4. Select **Connect**.
5. Add the URL and port for the repository.
6. If authentication is required, enable the **Use authentication** check box and enter the credentials for the repository.
7. Select **Connect**.

Result

The repository is connected. In the **Artifact Repository** section, the repository should show a connected status.

Disconnect an artifact repository

You can remove the connection to an artifact repository if it is no longer needed.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Connections** tab.

3. In the **Artifact Repository** section, select the menu at the top right.
4. Select **Disconnect**.
5. Select **Yes, disconnect artifact repository**.

Result

The repository is disconnected. In the **Artifact Repository** section, the repository should show a connected status.

Find more information

- [Manage software packages](#)

Manage software packages

NetApp delivers additional capabilities for Astra Control Center with software packages that you can download from the NetApp Support Site. After you connect Docker and artifact repositories, you can upload and import packages to add this functionality to Astra Control Center. You can use the CLI or the Astra Control Center web UI to manage software packages.

What you'll need

- Kubernetes cluster with Astra Control Center installed
- A connected Docker image repository to hold software package images. For more information, see [Manage repository connections](#).
- A connected artifact repository to hold software package binaries and artifacts. For more information, see [Manage repository connections](#).
- A software package from the NetApp Support Site

Upload a software package to the repositories

Astra Control Center references package images and artifacts in connected repositories. You can upload images and artifacts to the repositories using the CLI.

Steps

1. Download the software package from the NetApp Support Site, and save it on a machine that has the `kubectl` utility installed.
2. Extract the compressed package file, and change directory to the location of the Astra Control bundle file (for example, `acc.manifest.bundle.yaml`).
3. Push the package images to the Docker repository. Make the following substitutions:
 - Replace `BUNDLE_FILE` with the name of the Astra Control bundle file.
 - 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
```

4. Copy the package artifacts to the artifact repository. Replace `BUNDLE_FILE` with the name of the Astra Control bundle file, and `NETWORK_LOCATION` with the network location to copy the artifact files to:

```
kubectl astra packages copy-artifacts -m BUNDLE_FILE -n NETWORK_LOCATION
```

Add a software package

You can import software packages using an Astra Control Center bundle file. Doing this installs the package and makes the software available for Astra Control Center to use.

Add a software package using the Astra Control web UI

You can use the Astra Control Center web UI to add a software package that has been uploaded to the connected repositories.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Packages** tab.
3. Select the **Add** button.
4. In the file selection dialog, select the upload icon.
5. Choose an Astra Control bundle file, in `.yaml` format, to upload.
6. Select **Add**.

Result

If the bundle file is valid and the package images and artifacts are located in your connected repositories, the package is added to Astra Control Center. When the status in the **Status** column changes to **Available**, you can use the package. You can hover over the status for a package to get more information.



If one or more images or artifacts for a package are not found in your repository, an error message appears for that package.

Add a software package using the CLI

You can use the CLI to import a software package that you have uploaded to the connected repositories. To do this, you first need to record your Astra Control Center account ID and an API token.

Steps

1. Using a web browser, log in to the Astra Control Center web UI.
2. From the Dashboard, select the user icon at the top right.
3. Select **API access**.
4. Note the Account ID near the top of the screen.
5. Select **Generate API token**.
6. In the resulting dialog, select **Generate API token**.
7. Note the resulting token, and select **Close**.
In the CLI, change directories to the location of the `.yaml` bundle file in the extracted package contents.

8. Import the package using the bundle file, making the following substitutions:

- Replace `BUNDLE_FILE` with the name of the Astra Control bundle file.
- Replace `SERVER` with the DNS name of the Astra Control instance.
- Replace `ACCOUNT_ID` and `TOKEN` with the account ID and API token you recorded earlier.

```
kubectl astra packages import -m BUNDLE_FILE -u SERVER -a ACCOUNT_ID  
-k TOKEN
```

Result

If the bundle file is valid and the package images and artifacts are located in your connected repositories, the package is added to Astra Control Center.



If one or more images or artifacts for a package are not found in your repository, an error message appears for that package.

Remove a software package

You can use the Astra Control Center web UI to remove a software package that you previously imported in Astra Control Center.

Steps

1. In the **Manage Your Account** navigation area, select **Account**.
2. Select the **Packages** tab.

You can see the list of installed packages and their statuses on this page.

3. In the **Actions** column for the package, open the actions menu.
4. Select **Delete**.

Result

The package is deleted from Astra Control Center, but the images and artifacts for the package remain in your repositories.

Find more information

- [Manage repository connections](#)

Manage buckets

An object store bucket provider is essential if you want to back up your applications and persistent storage or if you want to clone applications across clusters. Using Astra Control Center, add an object store provider as your off-cluster, backup destination for your apps.

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

Use one of the following Amazon Simple Storage Service (S3) bucket providers:

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



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

A bucket can be in one of these states:

- pending: The bucket is scheduled for discovery.
- available: The bucket is available for use.
- removed: The bucket is not currently accessible.

For instructions on how to manage buckets using the Astra Control API, see the [Astra Automation and API information](#).

You can do these tasks related to managing buckets:

- [Add a bucket](#)
- [Edit a bucket](#)
- [Rotate or remove bucket credentials](#)
- [Remove a bucket](#)



S3 buckets in Astra Control Center do not report available capacity. Before backing up or cloning apps managed by Astra Control Center, check bucket information in the ONTAP or StorageGRID management system.

Edit a bucket

You can change the access credential information for a bucket and change whether a selected bucket is the default bucket.



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. See the [Release Notes](#).

Steps

1. From the left navigation, select **Buckets**.
2. From the Options menu in the **Actions** column, select **Edit**.
3. Change any information other than the bucket type.



You can't modify the bucket type.

4. Select **Update**.

Rotate or remove bucket credentials

Astra Control uses bucket credentials to gain access and provide secret keys for an S3 bucket so that Astra Control Center can communicate with the bucket.

Rotate bucket credentials

If you rotate credentials, rotate them during a maintenance window when no backups are in progress (scheduled or on-demand).

Steps to edit and rotate credentials

1. From the left navigation, select **Buckets**.
2. From the Options menu in the **Actions** column, select **Edit**.
3. Create the new credential.
4. Select **Update**.

Remove bucket credentials

You should remove bucket credentials only if new credentials have been applied to a bucket, or if the bucket is no longer actively used.



The first set of credentials that you add to Astra Control is always in use because Astra Control uses the credentials to authenticate the backup bucket. Do not remove these credentials if the bucket is in active use as this will lead to backup failures and backup unavailability.



If you do remove active bucket credentials, see [troubleshooting bucket credential removal](#).

For instructions on how to remove S3 credentials using the Astra Control API, see the [Astra Automation and API information](#).

Remove a bucket

You can remove a bucket that is no longer in use or is not healthy. You might want to do this to keep your object store configuration simple and up-to-date.



You cannot remove a default bucket. If you want to remove that bucket, first select another bucket as the default.

What you'll need

- You should check to ensure that there are no running or completed backups for this bucket before you begin.
- You should check to ensure that the bucket is not being used in any active protection policy.

If there are, you will not be able to continue.

Steps

1. From left navigation, select **Buckets**.
2. From the **Actions** menu, select **Remove**.



Astra Control ensures first that there are no schedule policies using the bucket for backups and that there are no active backups in the bucket you are about to remove.

3. Type "remove" to confirm the action.
4. Select **Yes, remove bucket**.

Find more information

- [Use the Astra Control API](#)

Manage the 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. You can monitor storage capacity and health details, including performance if Astra Control Center is connected to Cloud Insights.

For instructions on how to manage storage backends using the Astra Control API, see the [Astra Automation and API information](#).

You can complete the following tasks related to managing a storage backend:

- [Add a storage backend](#)
- [View storage backend details](#)
- [Unmanage a storage backend](#)
- [Update a storage backend license](#)
- [Add nodes to a storage backend cluster](#)
- [Remove a storage backend](#)

View storage backend details

You can view storage backend information from the Dashboard or from the Backends option.

View storage backend details from the Dashboard

Steps

1. From the left navigation, select **Dashboard**.
2. Review the Storage backend section that shows the state:
 - **Unhealthy**: The storage is not in an optimal state. This could be due to a latency issue or an app is degraded due to a container issue, for example.
 - **All healthy**: The storage has been managed and is in an optimal state.
 - **Discovered**: The storage has been discovered, but not managed by Astra Control.

View storage backend details from the Backends option

View information about the backend health, capacity, and performance (IOPS throughput and/or latency).

With a connection to Cloud Insights, you can see the volumes that the Kubernetes apps are using, which are

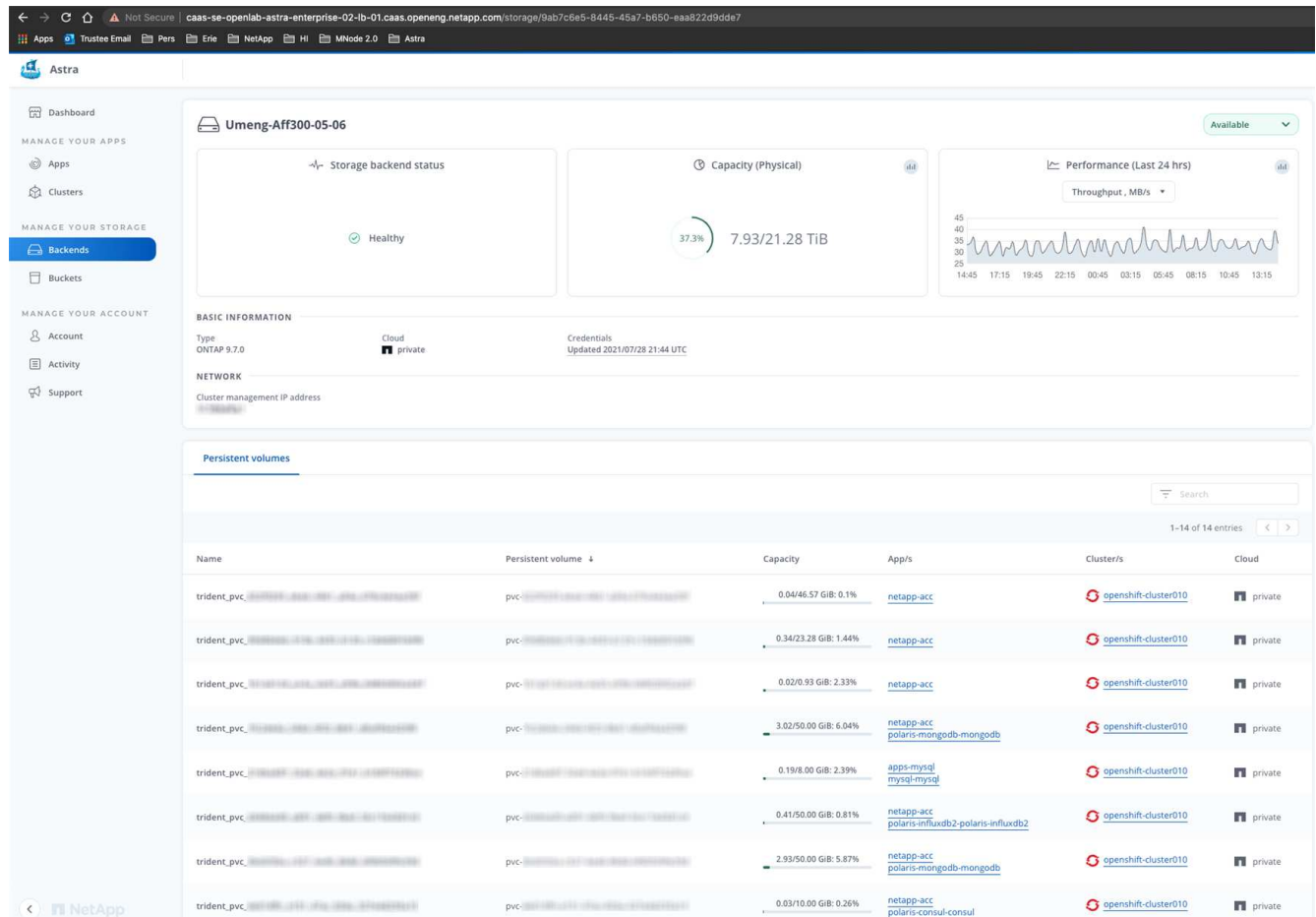
stored on a selected storage backend.

Steps

1. In the left navigation area, select **Backends**.
2. Select the storage backend.



If you connected to NetApp Cloud Insights, excerpts of data from Cloud Insights appear on the Backends page.



3. To go directly to Cloud Insights, select the **Cloud Insights** icon next to the metrics image.

Unmanage a storage backend

You can unmanage the backend.

Steps

1. From the left navigation, select **Backends**.
2. Select the storage backend.
3. From the Options menu in the **Actions** column, select **Unmanage**.
4. Type "unmanage" to confirm the action.
5. Select **Yes, unmanage storage backend**.

Remove a storage backend

You can remove a storage backend that is no longer in use. You might want to do this to keep your configuration simple and up-to-date.



If you are removing an Astra Data Store backend, it must not have been created by vCenter.

What you'll need

- Ensure that the storage backend is unmanaged.
- Ensure that the storage backend does not have any volumes associated with the Astra Data Store cluster.

Steps

1. From left navigation, select **Backends**.
2. If the backend is managed, unmanage it.
 - a. Select **Managed**.
 - b. Select the storage backend.
 - c. From the **Actions** option, select **Unmanage**.
 - d. Type "unmanage" to confirm the action.
 - e. Select **Yes, unmanage storage backend**.
3. Select **Discovered**.
 - a. Select the storage backend.
 - b. From the **Actions** option, select **Remove**.
 - c. Type "remove" to confirm the action.
 - d. Select **Yes, remove storage backend**.

Update a storage backend license

You can update the license for an Astra Data Store storage backend to support a larger deployment or enhanced features.

What you'll need

- A deployed and managed Astra Data Store storage backend
- An Astra Data Store license file (contact your NetApp sales representative to purchase an Astra Data Store license)

Steps

1. From the left navigation, select **Backends**.
2. Select the name of a storage backend.
3. Under **Basic Information**, you can see the type of license installed.

If you hover over the license information, a popup appears with more information, such as expiration and entitlement information.

4. Under **License**, select the edit icon next to the license name.
5. In the **Update license** page, do one of the following:

License status	Action
At least one license has been added to Astra Data Store.	Select a license from the list.
No licenses have been added to Astra Data Store.	<ol style="list-style-type: none"> Select the Add button. Select a license file to upload. Select Add to upload the license file.

- Select **Update**.

Add nodes to a storage backend cluster

You can add nodes to an Astra Data Store cluster, up to the number of nodes supported by the type of license installed for Astra Data Store.

What you'll need

- A deployed and licensed Astra Data Store storage backend
- You have added the Astra Data Store software package in Astra Control Center
- One or more new nodes to add to the cluster

Steps

- From the left navigation, select **Backends**.
- Select the name of a storage backend.
- Under Basic Information, you can see the number of nodes in this storage backend cluster.
- Under **Nodes**, select the edit icon next to the number of nodes.
- In the **Add nodes** page, enter information about the new node or nodes:
 - Assign a node label for each node.
 - Do one of the following:
 - If you want Astra Data Store to always use the maximum available number of nodes according to your license, enable the **Always use up to maximum number of nodes allowed** check box.
 - If you don't want Astra Data Store to always use the maximum available number of nodes, select the desired number of total nodes to use.
 - If you deployed Astra Data Store with Protection Domains enabled, assign the new node or nodes to Protection Domains.
- Select **Next**.
- Enter IP address and network information for each new node. Enter a single IP address for a single new node, or an IP address pool for multiple new nodes.

If Astra Data Store can use the IP addresses configured during deployment, you don't need to enter any IP address information.

- Select **Next**.
- Review the configuration for the new node or nodes.
- Select **Add nodes**.

Find more information

- [Use the Astra Control API](#)

Monitor and protect infrastructure

You can configure several optional settings to enhance your Astra Control Center experience. If the network where you're running Astra Control Center requires a proxy for connecting to the Internet (to upload support bundles to NetApp Support Site or establish a connection to Cloud Insights), you should configure a proxy server in Astra Control Center. To monitor and gain insight into your complete infrastructure, create a connection to NetApp Cloud Insights. To collect Kubernetes events from systems monitored by Astra Control Center, add a Fluentd connection.

Add a proxy server

If the network where you're running Astra Control Center requires a proxy for connecting to the Internet (to upload support bundles to NetApp Support Site or establish a connection to Cloud Insights), you should configure a proxy server in Astra Control Center.



Astra Control Center does not validate the details you enter for your proxy server. Ensure that you enter the correct values.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Connect** from the drop-down list to add a proxy server.



HTTP PROXY

Configure Astra Control to send traffic through a proxy server.

Disconnected



Connect

4. Enter the proxy server name or IP address and the proxy port number.
5. If your proxy server requires authentication, select the check box, and enter the username and password.
6. Select **Connect**.

Result

If the proxy information you entered was saved, the **HTTP Proxy** section of the **Account > Connections** page indicates that it is connected, and displays the server name.



Connected



HTTP PROXY ?

Server: proxy.example.com:8888

Authentication: Enabled

Edit proxy server settings

You can edit the proxy server settings.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Edit** from the drop-down list to edit the connection.
4. Edit the server details and authentication information.
5. Select **Save**.

Disable proxy server connection

You can disable the proxy server connection. You will be warned before you disable that potential disruption to other connections might occur.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Disconnect** from the drop-down list to disable the connection.
4. In the dialog box that opens, confirm the operation.

Connect to Cloud Insights

To monitor and gain insight into your complete infrastructure, connect NetApp Cloud Insights with your Astra Control Center instance. Cloud Insights is included in your Astra Control Center license.

Cloud Insights should be accessible from the network that Astra Control Center uses, or indirectly via a proxy server.

When Astra Control Center is connected to Cloud Insights, an Acquisition Unit pod gets created. This pod collects data from the storage backends that are managed by Astra Control Center and pushes it to Cloud Insights. This pod requires 8 GB RAM and 2 CPU cores.



After you enable the Cloud Insights connection, you can view throughput information on the **Backends** page as well as connect to Cloud Insights from here after selecting a storage backend. You can also find the information on the **Dashboard** in the Cluster section, and also connect to Cloud Insights from there.

What you'll need

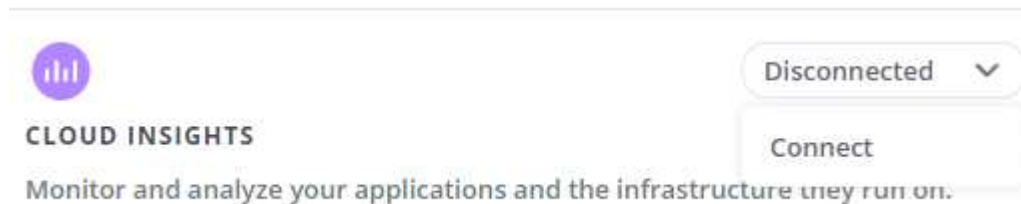
- An Astra Control Center account with **admin/owner** privileges.
- A valid Astra Control Center license.
- A proxy server if the network where you're running Astra Control Center requires a proxy for connecting to the Internet.



If you are new to Cloud Insights, familiarize yourself with the features and capabilities. See [Cloud Insights documentation](#).

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Connect** where it shows **Disconnected** in the drop-down list to add the connection.



4. Enter the Cloud Insights API tokens and the tenant URL. The tenant URL has the following format, as an example:

```
https://<environment-name>.c01.cloudinsights.netapp.com/
```

You get the tenant URL when you get the Cloud Insights license. If you do not have the tenant URL, see the [Cloud Insights documentation](#).

- a. To get the [API token](#), log in to your Cloud Insights tenant URL.
- b. In Cloud Insights, generate both a **Read/Write** and a **Read only** API Access token by clicking **Admin > API Access**.

Cloud Insights (Trial)

Tutorial 0% Complete

Getting Started

MONITOR & OPTIMIZE

HOME

DASHBOARDS

QUERIES

ALERTS

REPORTS

MANAGE

ADMIN

CLOUD SECURE

HELP

nmm95sx / Admin / API Access

API Access Tokens (4)

+ API Access Token

Bulk Actions

<input type="checkbox"/>	Name ↑	Description	Token	API Type	Permission
<input type="checkbox"/>	astra_...		...zBskB1	All Categories	Read/Write
<input type="checkbox"/>	astra_...		...xKOel_	All Categories	Read/Write
<input type="checkbox"/>	astra_...		...2_A6HP	All Categories	Read Only
<input type="checkbox"/>	astra_...		...8BTkYY	All Categories	Read/Write

- c. Copy the **Read only** key. You will need to paste it into the Astra Control Center window for enabling the Cloud Insights connection. For the Read API Access Token key permissions, select: Assets, Alerts, Acquisition Unit, and Data Collection.
- d. Copy the **Read/Write** key. You will need to paste it into the Astra Control Center **Connect Cloud Insights** window. For the Read/Write API Access Token key permissions, select: Assets, Data Ingestion, Log Ingestion, Acquisition Unit, and Data Collection.



We recommend that you generate a **Read only** key and a **Read/Write** key, and not use the same key for both purposes. By default, the token expiry period is set to one year. We recommend that you keep the default selection to give the token the maximum duration before it expires. If your token expires, the telemetry will stop.

- e. Paste the keys that you copied from Cloud Insights into Astra Control Center.

5. Select **Connect**.



After you select **Connect**, the status of the connection changes to **Pending** in the **Cloud Insights** section of the **Account > Connections** page. It can a few minutes for the connection to be enabled and the status to change to **Connected**.





To go back and forth easily between the Astra Control Center and Cloud Insights UIs, ensure that you are logged into both.

View data in Cloud Insights


If the connection was successful, the **Cloud Insights** section of the **Account > Connections** page indicates that it is connected, and displays the tenant URL. You can visit Cloud Insights to see data being successfully received and displayed.

EXTERNAL ?





Connected 

HTTP PROXY ?


Server: [proxy.example.com:8888](#) 

Authentication: Enabled




Connected 

CLOUD INSIGHTS ?

Tenant: [Cloud Insights](#) 

If the connection failed for some reason, the status shows **Failed**. You can find the reason for failure under **Notifications** at the top-right side of the UI.


Notifications Mark All as Read

 **Unable to connect to Cloud Insights** an hour ago

The Cloud Insights API token is invalid. Create a new API token in Cloud Insights and update Astra Control connection settings with the new token.

You can also find the same information under **Account > Notifications**.

From Astra Control Center, you can view throughput information on the **Backends** page as well as connect to Cloud Insights from here after selecting a storage backend.





 Backends

[+ Manage](#)

Search

★ Managed Q Discovered

1-1 of 1 entries < >

Name ↓	Status	Capacity	Throughput	Type	Actions
.06		7.67/21.28 TiB: 36%	 8.00 MB/s Throughput Last 24 hrs ● 5m ago: 8.00 MB/s ○ Min: 4.00 MB/s ● Max: 11.00 MB/s View in Cloud Insights 	ONTAP 9.7.0	Available 

To go directly to Cloud Insights, select the **Cloud Insights** icon next to the metrics image.

You can also find the information on the **Dashboard**.



After enabling the Cloud Insights connection, if you remove the backends that you added in Astra Control Center, the backends stop reporting to Cloud Insights.

Edit Cloud Insights connection

You can edit the Cloud Insights connection.



You can only edit the API keys. To change the Cloud Insights tenant URL, we recommended that you disconnect the Cloud Insights connection, and connect with the new URL.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Edit** from the drop-down list to edit the connection.
4. Edit the Cloud Insights connection settings.
5. Select **Save**.

Disable Cloud Insights connection

You can disable the Cloud Insights connection for a Kubernetes cluster managed by Astra Control Center. Disabling the Cloud Insights connection does not delete the telemetry data already uploaded to Cloud Insights.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Disconnect** from the drop-down list to disable the connection.
4. In the dialog box that opens, confirm the operation.
After you confirm the operation, on the **Account > Connections** page, the Cloud Insights status changes to **Pending**. It take a few minutes for the status to change to **Disconnected**.

Connect to Fluentd

You can send logs (Kubernetes events) from Astra Control Center to your Fluentd endpoint. The Fluentd connection is disabled by default.



Only the event logs from managed clusters are forwarded to Fluentd.

What you'll need

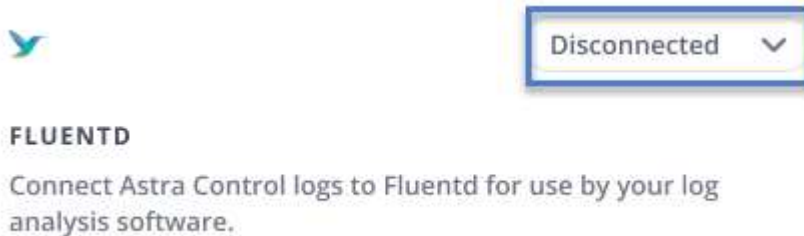
- An Astra Control Center account with **admin/owner** privileges.
- Astra Control Center installed and running on a Kubernetes cluster.



Astra Control Center does not validate the details you enter for your Fluentd server. Ensure that you enter the correct values.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Connect** from the drop-down list where it shows **Disconnected** to add the connection.



4. Enter the host IP address, the port number, and shared key for your Fluentd server.
5. Select **Connect**.

Result

If the details you entered for your Fluentd server were saved, the **Fluentd** section of the **Account > Connections** page indicates that it is connected. Now you can visit the Fluentd server that you connected and view the event logs.

If the connection failed for some reason, the status shows **Failed**. You can find the reason for failure under **Notifications** at the top-right side of the UI.

You can also find the same information under **Account > Notifications**.



If you are having trouble with log collection, you should log in to your worker node and ensure that your logs are available in `/var/log/containers/`.

Edit the Fluentd connection

You can edit the Fluentd connection to your Astra Control Center instance.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Edit** from the drop-down list to edit the connection.
4. Change the Fluentd endpoint settings.
5. Select **Save**.

Disable the Fluentd connection

You can disable the Fluentd connection to your Astra Control Center instance.

Steps

1. Log in to Astra Control Center using an account with **admin/owner** privilege.
2. Select **Account > Connections**.
3. Select **Disconnect** from the drop-down list to disable the connection.
4. In the dialog box that opens, confirm the operation.

Unmanage apps and clusters

Remove any apps or clusters that you no longer want to manage from Astra Control Center.

Unmanage an app

Stop managing apps that you no longer want to back up, snapshot, or clone from Astra Control Center.

- Any existing backups and snapshots will be deleted.
- Applications and data remain available.

Steps

1. From the left navigation bar, select **Applications**.
2. Select the check box for the apps that you no longer want to manage.
3. From the **Action** menu, select **Unmanage**.
4. Type "unmanage" to confirm.
5. Confirm that you want to unmanage the apps and then select **Yes, unmanage Application**.

Result

Astra Control Center stops managing the app.

Unmanage a cluster

Unmanage the cluster that you no longer want to manage from Astra Control Center.

- This action stops your cluster from being managed by Astra Control Center. It doesn't make any changes to the cluster's configuration and it doesn't delete the cluster.
- Trident won't be uninstalled from the cluster. [Learn how to uninstall Trident.](#)



Before you unmanage the cluster, you should unmanage the apps associated with the cluster.

Steps

1. From the left navigation bar, select **Clusters**.
2. Select the check box for the cluster that you no longer want to manage in Astra Control Center.
3. From the Options menu in the **Actions** column, select **Unmanage**.
4. Confirm that you want to unmanage the cluster and then select **Yes, unmanage cluster**.

Result

The status of the cluster changes to **Removing** and after that the cluster will be removed from the **Clusters** page, and it is no longer managed by Astra Control Center.



If Astra Control Center and Cloud Insights are not connected, unmanaging the cluster removes all the resources that were installed for sending telemetry data. **If Astra Control Center and Cloud Insights are connected**, unmanaging the cluster deletes only the `fluentbit` and `event-exporter` pods.

Upgrade Astra Control Center

To upgrade Astra Control Center, download the installation bundle from the NetApp Support Site and complete these instructions to upgrade the Astra Control Center components in your environment. You can use this procedure to upgrade Astra Control Center in internet-connected or air-gapped environments.

What you'll need

- [Before you begin upgrade, ensure your environment still meets the minimum requirements for Astra Control Center deployment.](#)
- Ensure all cluster operators are in a healthy state and available.

OpenShift example:

```
oc get clusteroperators
```

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

OpenShift example:

```
oc get apiservices
```

- Log out of your Astra Control Center.

About this task

The Astra Control Center upgrade process guides you through the following high-level steps:

- [Download the Astra Control Center bundle](#)
- [Unpack the bundle and change directory](#)
- [Add the images to your local registry](#)
- [Install the updated Astra Control Center operator](#)
- [Upgrade Astra Control Center](#)
- [Upgrade third-party services \(Optional\)](#)
- [Verify system status](#)
- [Set up ingress for load balancing](#)



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



Perform upgrades in a maintenance window when schedules, backups, and snapshots are not running.



Podman commands can be used in place of Docker commands if you are using Red Hat's Podman instead of Docker Engine.

Download the Astra Control Center bundle

1. Download the Astra Control Center upgrade bundle (`astra-control-center-[version].tar.gz`) from the [NetApp Support Site](#).
2. (Optional) Use the following command to verify the signature of the bundle:

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

Unpack the bundle and change directory

1. Extract the images:

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

2. Change to the Astra directory.

```
cd astra-control-center-[version]
```

Add the images to your local registry

1. Add the files in the Astra Control Center image directory to your local registry.



See a sample script for the automatic loading of images below.

- a. Log in to your Docker registry:

```
docker login [your_registry_path]
```

- b. Load the images into Docker.
- c. Tag the images.
- d. Push the images to your local registry.

```
export REGISTRY=[your_registry_path]
for astraImageFile in $(ls images/*.tar)
  # Load to local cache. And store the name of the loaded image
  trimming the 'Loaded images: '
  do astraImage=$(docker load --input ${astraImageFile} | sed
  's/Loaded image: //' )
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

Install the updated 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
```

- 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. Add the following values to the env section:

```
- name: ACCOP_HELM_UPGRADE_TIMEOUT  
  value: 300m
```

```

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"
            - name: ACCOP_HELM_UPGRADE_TIMEOUT
              value: 300m
          image: [your_registry_path]/acc-operator:[version x.y.z]
          imagePullPolicy: IfNotPresent
      imagePullSecrets: []

```

2. Install the updated Astra Control Center operator:

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

Sample response:

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

Upgrade Astra Control Center

1. Edit the Astra Control Center custom resource (CR) (`astra_control_center_min.yaml`) and change the Astra version (`astraVersion` inside of `Spec`) number to the latest:

```
kubectl edit acc -n [netapp-acc or custom namespace]
```



Your registry path must match the registry path where you pushed the images in a [previous step](#).

2. Add the following lines within `additionalValues` inside of `Spec` in the Astra Control Center CR:

```
additionalValues:
  nautilus:
    startupProbe:
      periodSeconds: 30
      failureThreshold: 600
```


3. Do one of the following:

- a. If you don't have your own IngressController or ingress and have been using the Astra Control Center with its Traefik gateway as a LoadBalancer type service and would like to continue with that setup, specify another field `ingressType` (if not already present) and set it to `AccTraefik`.

```
ingressType:AccTraefik
```

- b. If you want to switch to the default Astra Control Center generic ingress deployment, provide your own IngressController/Ingress setup (with TLS termination, etc.), open up a route to Astra Control Center, and set `ingressType` to `Generic`.

```
ingressType:Generic
```



If you omit the field, the process becomes the generic deployment. If you don't want the generic deployment, be sure to add the field.

4. (Optional) Verify that the pods terminate and become available again:

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

5. Wait for the Astra status conditions to indicate that the upgrade is complete and ready:

```
kubectl get -o yaml -n [netapp-acc or custom namespace]  
astracontrolcenters.astra.netapp.io astra
```

Response:

```
conditions:  
  - lastTransitionTime: "2021-10-25T18:49:26Z"  
    message: Astra is deployed  
    reason: Complete  
    status: "True"  
    type: Ready  
  - lastTransitionTime: "2021-10-25T18:49:26Z"  
    message: Upgrading succeeded.  
    reason: Complete  
    status: "False"  
    type: Upgrading
```

6. Log back in and verify that all managed clusters and apps are still present and protected.
7. If the operator did not update the Cert-manager, upgrade third-party services, next.

Upgrade third-party services (Optional)

The third-party services Traefik and Cert-manager are not upgraded during earlier upgrade steps. You can optionally upgrade them using the procedure described here or retain existing service versions if your system requires it.

- **Traefik:** By default, Astra Control Center manages the lifecycle of the Traefik deployment. Setting `externalTraefik` to `false` (default) indicates that no external Traefik exists in the system and Traefik is being installed and managed by Astra Control Center. In this case, `externalTraefik` is set to `false`.

On the other hand, if you have your own Traefik deployment, set `externalTraefik` to `true`. In this case, you maintain the deployment and Astra Control Center will not upgrade the CRDs, unless `shouldUpgrade` is set to `true`.

- **Cert-manager:** By default, Astra Control Center installs the cert-manager (and CRDs) unless you set `externalCertManager` to `true`. Set `shouldUpgrade` to `true` to have Astra Control Center upgrade the CRDs.

Traefik is upgraded if any of the following conditions are met:

- `externalTraefik: false`
OR
- `externalTraefik: true` AND `shouldUpgrade: true`.

Steps

1. Edit the `acc` CR:

```
kubectl edit acc -n [netapp-acc or custom namespace]
```

2. Change the `externalTraefik` field and the `shouldUpgrade` field to either `true` or `false` as needed.

```
crds:
  externalTraefik: false
  externalCertManager: false
  shouldUpgrade: false
```

Verify system status

1. Log in to Astra Control Center.
2. Verify that all your managed clusters and apps are still present and protected.

Set up ingress for load balancing

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

- Default upgrade uses the generic ingress deployment. In this case, you will also need to set up an ingress

controller or ingress resource.

- If you don't want an ingress controller and want to retain what you already have, set `ingressType` to `AccTraefik`.



For additional details about the service type of "LoadBalancer" and ingress, see [Requirements](#).

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

- Nginx ingress controller
- OpenShift ingress controller

What you'll need

- In the CR spec,
 - If `crd.externalTraefik` is present, it should be set to `false` OR
 - If `crd.externalTraefik` is `true`, `crd.shouldUpgrade` should also be `true`.
- 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 v1.21.

Steps for Nginx ingress controller

1. Use the existing secret `secure-testing-cert` or 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 for either a deprecated or a new schema:
 - a. For a 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: nginx
spec:
  tls:
    - hosts:
        - <ACC address>
      secretName: [tls secret name]
  rules:
    - host: [ACC address]
      http:
        paths:
          - backend:
              serviceName: traefik
              servicePort: 80
            pathType: ImplementationSpecific
```

b. For a new schema, follow this example:

```

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

```

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

```

Verify ingress set up

You can verify the ingress set up before you continue.

1. Ensure that Traefik has changed to `clusterIP` from `Loadbalancer`:

```

kubectl get service traefik -n [netapp-acc or custom namespace]

```

2. Verify routes in Traefik:

```
Kubectl get ingressroute ingressroutetls -n [netapp-acc or custom namespace]
-o yaml | grep "Host("
```



The result should be empty.

Uninstall Astra Control Center

You might need to remove Astra Control Center components if you are upgrading from a trial to a full version of the product. To remove Astra Control Center and the Astra Control Center Operator, run the commands described in this procedure in sequence.

If you have any issues with the uninstall, see [Troubleshooting uninstall issues](#).

What you'll need

- Use Astra Control Center UI to unmanage all [clusters](#).

Steps

1. Delete Astra Control Center. The following sample command is based upon a default installation. Modify the command if you made custom configurations.

```
kubectl delete -f astra_control_center_min.yaml -n netapp-acc
```

Result:

```
astracontrolcenter.astra.netapp.io "astra" deleted
```

2. Use the following command to delete the `netapp-acc` namespace:

```
kubectl delete ns netapp-acc
```

Result:

```
namespace "netapp-acc" deleted
```

3. Use the following command to delete Astra Control Center operator system components:

```
kubectl delete -f astra_control_center_operator_deploy.yaml
```

Result:

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

Troubleshooting uninstall issues

Use the following workarounds to address any problems you have with uninstalling Astra Control Center.

Uninstall of Astra Control Center fails to clean up the monitoring-operator pod on the managed cluster

If you did not unmanage your clusters before you uninstalled Astra Control Center, you can manually delete the pods in the netapp-monitoring namespace and the namespace with the following commands:

Steps

1. Delete acc-monitoring agent:

```
kubectl delete agents acc-monitoring -n netapp-monitoring
```

Result:

```
agent.monitoring.netapp.com "acc-monitoring" deleted
```

2. Delete the namespace:

```
kubectl delete ns netapp-monitoring
```

Result:

```
namespace "netapp-monitoring" deleted
```

3. Confirm resources removed:

```
kubectl get pods -n netapp-monitoring
```

Result:

```
No resources found in netapp-monitoring namespace.
```

4. Confirm monitoring agent removed:

```
kubectl get crd|grep agent
```

Sample result:

```
agents.monitoring.netapp.com                2021-07-21T06:08:13Z
```

5. Delete custom resource definition (CRD) information:

```
kubectl delete crds agents.monitoring.netapp.com
```

Result:

```
customresourcedefinition.apiextensions.k8s.io  
"agents.monitoring.netapp.com" deleted
```

Uninstall of Astra Control Center fails to clean up Traefik CRDs

You can manually delete the Traefik CRDs. CRDs are global resources, and deleting them might impact other applications on the cluster.

Steps

1. List Traefik CRDs installed on the cluster:

```
kubectl get crds |grep -E 'traefik'
```

Response

<code>ingressroutes.traefik.containo.us</code>	<code>2021-06-23T23:29:11Z</code>
<code>ingressroutetcps.traefik.containo.us</code>	<code>2021-06-23T23:29:11Z</code>
<code>ingressrouteudps.traefik.containo.us</code>	<code>2021-06-23T23:29:12Z</code>
<code>middlewares.traefik.containo.us</code>	<code>2021-06-23T23:29:12Z</code>
<code>middlewareetcps.traefik.containo.us</code>	<code>2021-06-23T23:29:12Z</code>
<code>serverstransports.traefik.containo.us</code>	<code>2021-06-23T23:29:13Z</code>
<code>tlsoptions.traefik.containo.us</code>	<code>2021-06-23T23:29:13Z</code>
<code>tlsstores.traefik.containo.us</code>	<code>2021-06-23T23:29:14Z</code>
<code>traefikservices.traefik.containo.us</code>	<code>2021-06-23T23:29:15Z</code>

2. Delete the CRDs:

```
kubectl delete crd ingressroutes.traefik.containo.us
ingressroutetcps.traefik.containo.us
ingressrouteudps.traefik.containo.us middlewares.traefik.containo.us
serverstransports.traefik.containo.us tlsoptions.traefik.containo.us
tlsstores.traefik.containo.us traefikservices.traefik.containo.us
middlewareetcps.traefik.containo.us
```

Find more information

- [Known issues for uninstall](#)

Automate with REST API

Automation using the Astra Control REST API

Astra Control has a REST API that enables you to directly access the Astra Control functionality using a programming language or utility such as Curl. You can also manage Astra Control deployments using Ansible and other automation technologies.

To set up and manage your Kubernetes apps, you can use either the Astra UI or the Astra Control API.

To learn more, go to the [Astra automation docs](#).

Deploy apps

Deploy Jenkins from a Helm chart

Learn how to deploy Jenkins from the [Bitnami Helm chart](#). After you deploy Jenkins on your cluster, you can register the application with Astra Control.

Jenkins is a validated app for Astra Control.

- [Learn the difference between a validated app and a standard app in Astra Control Service.](#)
- [Learn the difference between a validated app and a standard app in Astra Control Center.](#)

These instructions apply to both Astra Control Service and Astra Control Center.



Applications deployed from Google Marketplace have not been validated. Some users report issues with discovery and/or backup with Google Marketplace deployments of Postgres, MariaDB, and MySQL.

Requirements

- A cluster that has been added to Astra Control.



For Astra Control Center, you can add the cluster to Astra Control Center first or add the app first.

- Updated versions of Helm (version 3.2+) and Kubectl installed on a local machine with the proper kubeconfig for the cluster

Astra Control does not currently support the [Kubernetes plugin for Jenkins](#). You can run Jenkins in a Kubernetes cluster without the plugin. The plugin provides scalability to your Jenkins cluster.

Install Jenkins

Two important notes on this process:

- You must deploy your app after the cluster is added to Astra Control Service, not before. Astra Control Center will accept applications before or after the cluster is added to Astra Control Center.
- You must deploy the Helm chart in a namespace other than the default.

Steps

1. Add the Bitnami chart repo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Create the `jenkins` namespace and deploy Jenkins into it with the command:

```
Helm install <name> --namespace <namespace> --create-namespace --set  
persistence.storageClass=<storage_class>
```



If the volume size is changed, use Kibibyte (Ki), Mebibyte (Mi) or Gibibyte (Gi) units.

You need to define the storage class only in these situations:

- You are using Astra Control Service and you don't want to use the default storage class.
- You are using Astra Control Center and haven't yet imported the cluster into Astra Control Center. Or, you have imported the cluster, but don't want to use the default storage class.

Result

This does the following:

- Creates a namespace.
- Sets the correct storage class.

After the pods are online, you can manage the app with Astra Control. Astra Control enables you to manage an app at the namespace level or by using a helm label.

Deploy MariaDB from a Helm chart

Learn how to deploy MariaDB from the [Bitnami Helm chart](#). After you deploy MariaDB on your cluster, you can manage the application with Astra Control.

MariaDB is a validated app for Astra.

- [Learn the difference between a validated app and a standard app in Astra Control Service.](#)
- [Learn the difference between a validated app and a standard app in Astra Control Center.](#)

These instructions apply to both Astra Control Service and Astra Control Center.



Applications deployed from Google Marketplace have not been validated. Some users report issues with discovery and/or backup with Google Marketplace deployments of Postgres, MariaDB, and MySQL.

Requirements

- A cluster that has been added to Astra Control.



For Astra Control Center, you can add the cluster to Astra Control Center first or add the app first.

- Updated versions of Helm (version 3.2+) and Kubectl installed on a local machine with the proper kubeconfig for the cluster

Install MariaDB

Two important notes on this process:

- You must deploy your app after the cluster is added to Astra Control Service, not before. Astra Control Center will accept applications before or after the cluster is added to Astra Control Center.
- You must deploy the Helm chart in a namespace other than the default.

Steps

1. Add the Bitnami chart repo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Deploy MariaDB with the command:

```
Helm install <name> --namespace <namespace> --create-namespace --set  
persistence.storageClass=<storage_class>
```



If the volume size is changed, use Kibibyte (Ki), Mebibyte (Mi) or Gibibyte (Gi) units.

You need to define the storage class only in these situations:

- You are using Astra Control Service and you don't want to use the default storage class.
- You are using Astra Control Center and haven't yet imported the cluster into Astra Control Center. Or, you have imported the cluster, but don't want to use the default storage class.

Result

This does the following:

- Creates a namespace.
- Deploys MariaDB on the namespace.
- Creates a database.



This method of setting the password at deployment is insecure. We do not recommend this for a production environment.

After the pods are online, you can manage the app with Astra Control. Astra Control enables you to manage an app at the namespace level or by using a helm label.

Deploy MySQL from a Helm chart

Learn how to deploy MySQL from the [Bitnami Helm chart](#). After you deploy MySQL on your Kubernetes cluster, you can manage the application with Astra Control.

MySQL is a validated app for Astra Control.

- [Learn the difference between a validated app and a standard app in Astra Control Service.](#)

- [Learn the difference between a validated app and a standard app in Astra Control Center.](#)

These instructions apply to both Astra Control Service and Astra Control Center.



Applications deployed from Google Marketplace have not been validated. Some users report issues with discovery and/or backup with Google Marketplace deployments of Postgres, MariaDB, and MySQL.

Requirements

- A cluster that has been added to Astra Control.



For Astra Control Center, you can add the cluster to Astra Control Center first or add the app first.

- Updated versions of Helm (version 3.2+) and Kubectl installed on a local machine with the proper kubeconfig for the cluster

Install MySQL

Two important notes on this process:

- You must deploy your app after the cluster is added to Astra Control Service, not before. Astra Control Center will accept applications before or after the cluster is added to Astra Control Center.
- We recommend that you deploy the Helm chart in a namespace other than the default.

Steps

1. Add the Bitnami chart repo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Deploy MySQL with the command:

```
Helm install <name> --namespace <namespace> --create-namespace --set  
persistence.storageClass=<storage_class>
```



If the volume size is changed, use Kibibyte (Ki), Mebibyte (Mi) or Gibibyte (Gi) units.

You need to define the storage class only in these situations:

- You are using Astra Control Service and you don't want to use the default storage class.
- You are using Astra Control Center and haven't yet imported the cluster into Astra Control Center. Or, you have imported the cluster, but don't want to use the default storage class.

Result

This does the following:

- Creates a namespace.
- Deploys MySQL on the namespace.

After the pods are online, you can manage the app with Astra Control. Astra Control allows you to manage an app with its name, at the namespace level, or by using a helm label.

Deploy Postgres from a Helm chart

Learn how to deploy Postgres from the [Bitnami Helm chart](#). After you deploy Postgres on your cluster, you can register the application with Astra Control.

Postgres is a validated app for Astra.

- [Learn the difference between a validated app and a standard app in Astra Control Service.](#)
- [Learn the difference between a validated app and a standard app in Astra Control Center.](#)

These instructions apply to both Astra Control Service and Astra Control Center.



Applications deployed from Google Marketplace have not been validated. Some users report issues with discovery and/or backup with Google Marketplace deployments of Postgres, MariaDB, and MySQL.

Requirements

- A cluster that has been added to Astra Control.



For Astra Control Center, you can add the cluster to Astra Control Center first or add the app first.

- Updated versions of Helm (version 3.2+) and Kubectl installed on a local machine with the proper kubeconfig for the cluster

Install Postgres

Two important notes on this process:

- You must deploy your app after the cluster is added to Astra Control Service, not before. Astra Control Center will accept applications before or after the cluster is added to Astra Control Center.
- You must deploy the Helm chart in a namespace other than the default.

Steps

1. Add the Bitnami chart repo:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Deploy Postgres with the command:

```
Helm install <name> --namespace <namespace> --create-namespace --set  
persistence.storageClass=<storage_class>
```



If the volume size is changed, use Kibibyte (Ki), Mebibyte (Mi) or Gibibyte (Gi) units.

You need to define the storage class only in these situations:

- You are using Astra Control Service and you don't want to use the default storage class.
- You are using Astra Control Center and haven't yet imported the cluster into Astra Control Center. Or, you have imported the cluster, but don't want to use the default storage class.

Result

This does the following:

- Creates a namespace.
- Deploys Postgres on the namespace.

After the pods are online, you can manage the app with Astra Control. Astra Control enables you to manage an app at the namespace level or by using a helm label.

Knowledge and support

Troubleshooting

Learn how to work around some common problems you might encounter.

https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra

Find more information

- [How to upload a file to NetApp \(login required\)](#)
- [How to manually upload a file to NetApp \(login required\)](#)

Get help

NetApp provides support for Astra Control in a variety of ways. Extensive free self-support options are available 24x7, such as knowledgebase (KB) articles and a Slack channel. Your Astra Control account includes remote technical support via web ticketing.



If you have an evaluation license for Astra Control Center, you can get technical support. However, case creation via NetApp Support Site (NSS) is not available. You can get in touch with Support via the feedback option or use the Slack channel for self service.

You must first [activate support for your NetApp serial number](#) in order to use these non self-service support options. A NetApp Support Site (NSS) SSO account is required for chat and web ticketing along with case management.

Self-support options

You can access support options from the Astra Control Center UI by selecting the **Support** tab from the main menu.

These options are available for free, 24x7:

- **Knowledge base (login required):** Search for articles, FAQs, or Break Fix information related to Astra Control.
- **Documentation center:** This is the doc site that you're currently viewing.
- **Get help via Slack:** Go to the containers channel in thePub workspace to connect with peers and experts.
- **Create a support case:** Generate support bundles to provide to NetApp Support for troubleshooting.
- **Give feedback about Astra Control:** Send an email to astra.feedback@netapp.com to let us know your thoughts, ideas, or concerns.

Enable daily scheduled support bundle upload to NetApp Support

During Astra Control Center installation, if you specify `enrolled: true` for `autoSupport` in the Astra Control Center Custom Resource Definition (CRD) file (`astra_control_center_min.yaml`), daily support bundles are automatically uploaded to the [NetApp Support Site](#).

Generate support bundle to provide to NetApp Support

Astra Control Center enables the admin user to generate bundles, which include information useful to NetApp Support, including logs, events for all the components of the Astra deployment, metrics, and topology information about the clusters and apps under management. If you are connected to the Internet, you can upload support bundles to NetApp Support Site (NSS) directly from the Astra Control Center UI.



The time taken by Astra Control Center to generate the bundle depends on the size of your Astra Control Center installation as well as the parameters of the requested support bundle. The time duration that you specified when requesting a support bundle dictates the time it takes for the bundle to be generated (for example, a shorter time period results in faster bundle generation).

Before you begin

Determine whether a proxy connection will be required to upload bundles to NSS. If a proxy connection is needed, verify that Astra Control Center has been configured to use a proxy server.

1. Select **Accounts > Connections**.
2. Check the proxy settings in **Connection settings**.

Steps

1. Create a case on the NSS portal using the license serial number listed on the **Support** page of the Astra Control Center UI.
2. Perform the following steps for generating the support bundle by using the Astra Control Center UI:
 - a. On the **Support** page, in the Support bundle tile, select **Generate**.
 - b. In the **Generate a Support Bundle** window, select the timeframe.

You can choose between quick or custom timeframes.



You can choose a custom date range as well as specify a custom time period during the date range.

- c. After you make the selections, select **Confirm**.
- d. Select the **Upload the bundle to the NetApp Support Site when generated** check box.
- e. Select **Generate Bundle**.

When the support bundle is ready, a notification appears on the **Accounts > Notification** page in the Alerts area, on the **Activity** page, and also in the notifications list (accessible by selecting the icon in the top-right side of the UI).

If the generation failed, an icon appears on the Generate Bundle page. Select the icon to see the message.



The notifications icon at the top-right side of the UI provides information about events related to the support bundle, such as when the bundle is successfully created, when the bundle creation fails, when the bundle could not be uploaded, when the bundle could not be downloaded, and so on.

If you have an air-gapped installation

If you have an air-gapped installation, perform the following steps after the Support bundle is generated. When the bundle is available for download, the Download icon appears next to **Generate** in the **Support Bundles** section of the **Support** page.

Steps

1. Select the Download icon to download the bundle locally.
2. Manually upload the bundle to NSS.

You can use one of the following methods to do this:

- Use [NetApp Authenticated File Upload \(login required\)](#).
- Attach the bundle to the case directly on NSS.
- Use NetApp Active IQ.

Find more information

- [How to upload a file to NetApp \(login required\)](#)
- [How to manually upload a file to NetApp \(login required\)](#)

Earlier versions of Astra Control Center documentation

Documentation for previous releases is available.

- [Astra Control Center 21.12 documentation](#)
- [Astra Control Center 21.08 documentation](#)

Legal notices

Legal notices provide access to copyright statements, trademarks, patents, and more.

Copyright

<http://www.netapp.com/us/legal/copyright.aspx>

Trademarks

NETAPP, the NETAPP logo, and the marks listed on the NetApp Trademarks page are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

<http://www.netapp.com/us/legal/netapptmlist.aspx>

Patents

A current list of NetApp owned patents can be found at:

<https://www.netapp.com/us/media/patents-page.pdf>

Privacy policy

<https://www.netapp.com/us/legal/privacypolicy/index.aspx>

Open source

Notice files provide information about third-party copyright and licenses used in NetApp software.

- [Notice for Astra Control Center](#)
- [Notice for Astra Data Store](#)

Astra Control API license

<https://docs.netapp.com/us-en/astra-automation/media/astra-api-license.pdf>

Copyright Information

Copyright © 2022 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.