Astra Control Center 24.02 documentation

Astra Control Center

NetApp
April 25, 2024

Table of Contents

Astra Control Center 24.02 documentation

Release notes

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

- What's in this release of Astra Control Center
- Known issues
- Known limitations

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 release of Astra Control Center.

15 March 2024 (24.02.0)

New features and support

- **Deploy Astra Control Center without a private registry**: You no longer need to push Astra Control Center images to a private registry or use one as part of your Astra Control environment.
- **Minor bug fixes**

Known issues and limitations

- Known issues for this release
- Known limitations for this release

(Tech preview) Declarative Kubernetes workflows

This Astra Control Center release contains declarative Kubernetes functionality that enables you to perform data management from a native Kubernetes custom resource (CR).

After you install the Astra Connector on the cluster you want to manage, you'll be able to perform the following CR-based cluster operations in the UI or from a CR:

- Define an application using a custom resource
- Define the bucket
- Protect an entire cluster
- Back up your application
- Create a snapshot
- Create schedules for snapshots or backups
- Restore an application from a snapshot or backup

7 November 2023 (23.10.0)

New features and support

- **Backup and restore capabilities for applications with ontap-nas-economy driver-backed storage backends**: Enable backup and restore operations for `ontap-nas-economy` with some simple steps.

- **Immutable Backups**: Astra Control now supports unalterable, read-only backups as an additional security layer against malware and other threats.
- **Introducing Astra Control Provisioner**

  With the 23.10 release, Astra Control introduces a new software component called Astra Control Provisioner that will be available to all licensed Astra Control users. Astra Control Provisioner provides access to a superset of advanced management and storage provisioning features beyond those that Astra Trident provides. These features are available to all Astra Control customers at no additional cost.

  - **Get started with Astra Control Provisioner**
    You can enable Astra Control Provisioner if you have installed and configured your environment to use Astra Trident 23.10.
  - **Astra Control Provisioner functionality**

    The following features are available with the Astra Control Provisioner 23.10 release:

    - **Enhanced storage backend security with Kerberos 5 encryption**: You can improve storage security by enabling encryption for the traffic between your managed cluster and the storage backend. Astra Control Provisioner supports Kerberos 5 encryption over NFSv4.1 connections from Red Hat OpenShift clusters to Azure NetApp Files and on-premises ONTAP volumes
    - **Recover data using a snapshot**: Astra Control Provisioner provides rapid, in-place volume restoration from a snapshot using the `TridentActionSnapshotRestore` (TASR) CR.
    - **SnapMirror enhancements**: Use the app replication feature in environments where Astra Control does not have direct connectivity to an ONTAP cluster or access to ONTAP credentials. This feature allows you to use replication without having to manage a storage backend or its credentials in Astra Control.
    - **Backup and restore capabilities for applications with `ontap-nas-economy` driver-backed storage backends**: As described above.
- **Support for managing applications that use NVMe/TCP storage**
  Astra Control can now manage applications backed by persistent volumes that are connected using NVMe/TCP.
- **Execution hooks turned off by default**: Beginning with this release, execution hooks functionality can be enabled or disabled for additional security (it is disabled by default). If you have not yet created execution hooks for use with Astra Control, you need to enable the execution hooks feature to begin creating hooks. If you created execution hooks prior to this release, the execution hooks functionality stays enabled and you can use hooks as you would normally.

Known issues and limitations

- Known issues for this release
- Known limitations for this release

31 July 2023 (23.07.0)

New features and support

- Support for using NetApp MetroCluster in a stretch configuration as a storage backend
- Support for using Longhorn as a storage backend
- Applications can now be replicated between ONTAP backends from the same Kubernetes cluster
- Astra Control Center now supports 'userPrincipalName' as an alternative login attribute for remote (LDAP)

users

- New execution hook type 'post-failover' can be run after replication failover with Astra Control Center
- Clone workflows now support live clones only (the current state of managed application). To clone from a snapshot or backup, use the restore workflow.

Known issues and limitations

- Known issues for this release
- Known limitations for this release

18 May 2023 (23.04.2)

This patch release (23.04.2) for Astra Control Center (23.04.0) provides support for Kubernetes CSI external snapshotter v6.1.0 and fixes the following:

- A bug with in-place application restore when using execution hooks
- Connection issues with the bucket service

25 April 2023 (23.04.0)

New features and support

- 90-day evaluation license enabled by default for new Astra Control Center installations
- Enhanced execution hooks functionality with additional filtering options
- Execution hooks can now be run after replication failover with Astra Control Center
- Support for migrating volumes from the 'ontap-nas-economy storage' class to the 'ontap-nas' storage class
- Support for including or excluding application resources during restore operations
- Support for managing data-only applications

Known issues and limitations

- Known issues for this release
- Known limitations for this release

22 November 2022 (22.11.0)

New features and support

- Support for applications that span across multiple namespaces
- Support for including cluster resources in an application definition
- Enhanced LDAP authentication with role-based access control (RBAC) integration
- Added support for Kubernetes 1.25 and Pod Security Admission (PSA)
- Enhanced progress reporting for your backup, restore, and clone operations

Known issues and limitations

- Known issues for this release
- Known limitations for this release

8 September 2022 (22.08.1)

This patch release (22.08.1) for Astra Control Center (22.08.0) fixes minor bugs in app replication using NetApp SnapMirror.

10 August 2022 (22.08.0)

New features and support

- App replication using NetApp SnapMirror technology
- Improved app management workflow
- Enhanced provide-your-own execution hooks functionality

> The NetApp provided default pre- and post-snapshot execution hooks for specific applications have been removed in this release. If you upgrade to this release and do not provide your own execution hooks for snapshots, Astra Control will take crash-consistent snapshots only. Visit the NetApp Verda GitHub repository for sample execution hook scripts that you can modify to fit your environment.

- Support for VMware Tanzu Kubernetes Grid Integrated Edition (TKGI)
- Support for Google Anthos
- LDAP configuration (via Astra Control API)

Known issues and limitations

- Known issues for this release
- Known limitations for this release

26 April 2022 (22.04.0)

New features and support

- 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 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 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 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
- 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:

- App backups and snapshots fail if the volumesnapshotclass is added after a cluster is managed
- Managing a cluster with Astra Control Center fails when kubeconfig file contains more than one context
- App data management operations fail with Internal Service Error (500) when Astra Trident is offline
- In-place restore operations to ontap-nas-economy storage classes fail
- Restoring from a backup when using Kerberos in-flight encryption can fail
- Backup data remains in bucket after deletion for buckets with expired retention policy

App backups and snapshots fail if the volumesnapshotclass is added after a cluster is managed

Backups and snapshots fail with a `UI 500 error` in this scenario. As a workaround, refresh the app list.

Managing a cluster with Astra Control Center fails when 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.

In-place restore operations to ontap-nas-economy storage classes fail

If you perform an in-place restore of an application (restore the app to its original namespace), and the app's storage class uses the `ontap-nas-economy` driver, the restore operation can fail if the snapshot directory is not hidden. Before restoring in-place, follow the instructions in Enable backup and restore for ontap-nas-economy operations to hide the snapshot directory.

Restoring from a backup when using Kerberos in-flight encryption can fail

When you restore an application from a backup to a storage backend that is using Kerberos in-flight encryption, the restore operation can fail. This issue does not affect restoring from a snapshot or replicating the application data using NetApp SnapMirror.

> When using Kerberos in-flight encryption with NFSv4 volumes, ensure that the NFSv4 volumes are using the correct settings. Refer to the NetApp NFSv4 Domain Configuration section (page 13) of the NetApp NFSv4 Enhancements and Best Practices Guide.

Backup data remains in bucket after deletion for buckets with expired retention policy

If you delete an app's immutable backup after the bucket's retention policy has expired, the backup is deleted from Astra Control but not from the bucket. This issue will be fixed in an upcoming release.

Find more information

- 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
- Restrictive role constraints can be ignored for resources on non-connector clusters

App management limitations

- Multiple applications in a single namespace cannot be restored collectively to a different namespace
- Astra Control does not support apps that use multiple storage classes per namespace
- Astra Control does not automatically assign default buckets for cloud instances
- 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
- Snapshots might fail for Kubernetes 1.25 or later clusters with certain snapshot controller versions
- Backups and snapshots might not be retained during removal of an Astra Control Center instance

General limitations

- LDAP user and group 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
- The Activity page displays up to 100000 events
- SnapMirror does not support applications using NVMe over TCP for storage backends

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 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 clone or restore operation creates a new namespace, 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.

Restrictive role constraints can be ignored for resources on non-connector clusters

- **If the resources being accessed belong to clusters that have the latest Astra Connector installed**: When a user is assigned multiple roles through LDAP group membership, the constraints from the roles are combined. For example, if a user with a local Viewer role joins three groups that are bound to the Member role, the user now has Viewer role access to the original resources as well as Member role access to the resources gained through group membership.
- **If the resources being accessed belong to clusters that do not have Astra Connector installed**: When a user is assigned multiple roles through LDAP group membership, the constraints from the most permissive role are the only ones that take effect.

Multiple applications in a single namespace cannot be restored collectively to a different namespace

If you manage multiple applications in a single namespace (by creating multiple app definitions in Astra Control), you cannot restore all of the applications to a different single namespace. You need to restore each application to its own separate namespace.

Astra Control does not support apps that use multiple storage classes per namespace

Astra Control supports apps that use a single storage class per namespace. When you add an app to a namespace, be sure the app has the same storage class as other apps in the namespace.

Astra Control does not automatically assign default buckets for cloud instances

Astra Control does not automatically assign a default bucket for any cloud instance. You need to manually set a default bucket for a cloud instance. If a default bucket is not set, you won't be able to perform app clone operations between two clusters.

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

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.

During clone operations, apps that need an IngressClass resource or webhooks to function properly must not have those resources already defined on the destination cluster.

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, refer to Astra Control Center requirements.

Snapshots might fail for Kubernetes 1.25 or later clusters with certain snapshot controller versions

Snapshots for Kubernetes clusters running version 1.25 or later can fail if version v1beta1 of the snapshot controller APIs are installed on the cluster.

As a workaround, do the following when upgrading existing Kubernetes 1.25 or later installations:

1. Remove any existing Snapshot CRDs and any existing snapshot controller.
2. Uninstall Astra Trident.
3. Install the snapshot CRDs and the snapshot controller.
4. Install the latest Astra Trident version.
5. Create a VolumeSnapshotClass.

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.

LDAP user and group limitations

Astra Control Center supports up to 5,000 remote groups and 10,000 remote users.

Astra Control does not support an LDAP entity (user or group) that has a DN containing an RDN with a trailing '\' or trailing space.

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.

The Activity page displays up to 100000 events

The Astra Control Activity page can display up to 100,000 events. To view all logged events, retrieve the events using the Astra Control API.

SnapMirror does not support applications using NVMe over TCP for storage backends

Astra Control Center does not support NetApp SnapMirror replication for storage backends that are using the NVMe over TCP protocol.

Find more information

- Known issues

Get started

Learn about Astra Control

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

Features

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

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

Deployment models

Astra Control is available in two deployment models:

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

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

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

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

How Astra Control Service works

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

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

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

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

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

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

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

How Astra Control Center works

Astra Control Center runs locally in your own private cloud.

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

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

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

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

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

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

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

Center.

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

Learn more about how to set up Astra Control Center.

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

For more information

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

Astra Control Center requirements

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

Supported host cluster Kubernetes environments

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

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

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

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

Host cluster resource requirements

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

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

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

Service mesh requirements

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

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

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

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

Astra Trident

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

Astra Control Provisioner

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

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

ONTAP configuration with Astra Trident

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

Storage backends

Ensure that you have a supported backend with sufficient capacity.

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

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

ONTAP licenses

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

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

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

NetApp MetroCluster

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

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

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

- SAN
- NAS

Astra Control Center license

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

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

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

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

Networking requirements

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

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

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

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

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

Ingress for on-premises Kubernetes clusters

You can choose the type of network ingress Astra Control Center uses. By default, Astra Control Center deploys the Astra Control Center gateway (service/traefik) as a cluster-wide resource. Astra Control Center also supports using a service load balancer, if they are permitted in your environment. If you would rather use a service load balancer and you don't already have one configured, you can use the MetalLB load balancer to automatically assign an external IP address to the service. In the internal DNS server configuration, you should point the chosen DNS name for Astra Control Center to the load-balanced IP address.

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

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

Google Anthos ingress requirements

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

Supported web browsers

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

Additional requirements for application clusters

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

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

What's next

View the quick start overview.

Quick start for Astra Control Center

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

1 Review Kubernetes cluster requirements

Ensure that your environment meets these requirements:

Kubernetes cluster

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

Storage integration

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

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

ONTAP credentials

- Configure ONTAP credentials

2 Download and install Astra Control Center

Complete these installation tasks:

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

3 Complete some initial setup tasks

Complete some basic tasks to get started:

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

4 Use Astra Control Center

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

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

For more information

- Use the Astra Control API
- Upgrade Astra Control Center

- Get help with Astra Control

Installation overview

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

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

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

- Configure Astra Control Center after installation

Install Astra Control Center using the standard process

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

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

Before you begin

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

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

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

```
kubectl get apiservices
```

- **Ensure a routable FQDN**: The Astra FQDN you plan to use can be routed to the cluster. This means that you either have a DNS entry in your internal DNS server or you are using a core URL route that is already registered.
- **Configure cert manager**: If a cert manager already exists in the cluster, you need to perform some prerequisite steps so that Astra Control Center does not attempt to install its own cert manager. By default, Astra Control Center installs its own cert manager during installation.
- **(ONTAP SAN driver only) Enable multipath**: If you are using an ONTAP SAN driver, be sure that multipath is enabled on all your Kubernetes clusters.

You should also consider the following:

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

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

Astra Control Provisioner, from the NetApp image registry.

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

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

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

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

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

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

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

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

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

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

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

Steps

To install Astra Control Center, do the following steps:

- Download and extract Astra Control Center
- Complete additional steps if you use a local registry
- Set up namespace and secret for registries with auth requirements
- Install the Astra Control Center operator
- Configure Astra Control Center
- Complete Astra Control Center and operator installation
- Verify system status
- Set up ingress for load balancing
- Log in to the Astra Control Center UI

Do not delete the Astra Control Center operator (for example, `kubectl delete -f astra_control_center_operator_deploy.yaml`) at any time during Astra Control Center installation or operation to avoid deleting pods.

Download and extract Astra Control Center

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

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

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

Astra Control image registry

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

NetApp Support Site

1. Download the bundle containing Astra Control Center (`astra-control-center-[version].tar.gz`) from the Astra Control Center downloads page.
2. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (`astra-control-center-certs-[version].tar.gz`) to verify the signature of the bundle.

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

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

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

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

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

Complete additional steps if you use a local registry

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

Install the NetApp Astra kubectl plugin

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

Before you begin

NetApp provides plugin binaries for different CPU architectures and operating systems. You need to know which CPU and operating system you have before you perform this task.

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

Steps

1. List the available NetApp Astra kubectl plugin binaries:

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

   ```
   ls kubectl-astra/
   ```

2. Move the file you need for your operating system and CPU architecture into the current path and rename it to `kubectl-astra`:

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

Add the images to your registry

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

Docker

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

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

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

- Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.bundle.yaml`).
- Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, "https://<docker-registry>".
- Replace <MY_REGISTRY_USER> with the user name.
- Replace <MY_REGISTRY_TOKEN> with an authorized token for the registry.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

Podman

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

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

b. Log in to your registry:

```
podman login <YOUR_REGISTRY>
```

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

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

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

2. Change the directory:

```
cd manifests
```

Set up namespace and secret for registries with auth requirements

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

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

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

2. If you use a registry that requires authentication, you need to do the following:
   a. Create the `netapp-acc-operator` namespace:

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

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

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

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

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

If you delete the namespace after the secret is generated, recreate the namespace and then regenerate the secret for the namespace.

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

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

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

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

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

Install the Astra Control Center operator

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

   a. Open the Astra Control Center operator deployment YAML:

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   ⓘ An annotated sample YAML follows these steps.

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

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

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

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

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

```
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: ASTRA_IMAGE_REGISTRY/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
```

```
            memory: 75Mi
          securityContext:
            allowPrivilegeEscalation: false
        imagePullSecrets: []
        securityContext:
          runAsUser: 65532
        terminationGracePeriodSeconds: 10
```

2. Install the Astra Control Center operator:

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

Expand for sample response:

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

3. Verify pods are running:

```
kubectl get pods -n netapp-acc-operator
```

Configure Astra Control Center

1. Edit the Astra Control Center custom resource (CR) file (`astra_control_center.yaml`) to make

account, support, registry, and other necessary configurations:

```
vim astra_control_center.yaml
```

An annotated sample YAML follows these steps.

2. Modify or confirm the following settings:

accountName

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

astraVersion

Setting	Guidance	Type	Example
`astraVersion`	The version of Astra Control Center to deploy. No action is needed for this setting as the value will be pre-populated.	string	`24.02.0-69`

astraAddress

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

autoSupport

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

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

email

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

firstName

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

LastName

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

imageRegistry

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

Setting	Use	Guidance	Type	Example
`imageRegistry.name`	Required	The name of the Astra Control image registry that hosts all images required to deploy Astra Control Center. The value will be pre-populated, and no action is required unless you configured a local registry. For a local registry, replace this existing value with the name of the image registry where you pushed the images in the previous step. Do not use `http://` or `https://` in the registry name.	string	`cr.astra.netapp.io` (default) `example.registry.com/astra` (local registry example)

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

storageClass

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

volumeReclaimPolicy

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

ingressType

Setting	Guidance	Type	Options
`ingressType`	Use one of the following ingress types: **Generic** (`ingressType: "Generic"`) (Default) Use this option when you have another ingress controller in use or would prefer to use your own ingress controller. After Astra Control Center is deployed, you'll need to configure the ingress controller to expose Astra Control Center with a URL. IMPORTANT: If you intend to use a service mesh with Astra Control Center, you must select `Generic` as ingress type and set up your own ingress controller. **AccTraefik** (`ingressType: "AccTraefik"`) Use this option when you would prefer not to configure an ingress controller. This deploys the Astra Control Center `traefik` gateway as a Kubernetes LoadBalancer type service. Astra Control Center uses a service of the type "LoadBalancer" (`svc/traefik` in the Astra Control Center namespace), and requires that it be assigned an accessible external IP address. If load balancers are permitted in your environment and you don't already have one	string	• `Generic` (this is the default value) • `AccTraefik`

scaleSize

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

astraResourcesScaler

Setting	Guidance	Type	Options
`astraResourcesScaler`	Scaling options for AstraControlCenter Resource limits. By default, Astra Control Center deploys with resource requests set for most of the components within Astra. This configuration allows the Astra Control Center software stack to perform better in environments under increased application load and scale. However, in scenarios using smaller development or test clusters, the CR field `astraResourcesScalar` may be set to `Off`. This disables resource requests and allows for deployment on smaller clusters.	string	• `Default` (This is the default value) • `Off`

additionalValues

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

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

crds

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

Setting	Guidance	Type	Example
`crds.externalCertManager`	If you use an external cert manager, change `externalCertManager` to `true`. The default `false` causes Astra Control Center to install its own cert manager CRDs during installation. CRDs are cluster-wide objects and installing them might have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.	Boolean	`False` (this value is the default)
`crds.externalTraefik`	By default, Astra Control Center will install required Traefik CRDs. CRDs are cluster-wide objects and installing them might have an impact on other parts of the cluster. You can use this flag to signal to Astra Control Center that these CRDs will be installed and managed by the cluster administrator outside of Astra Control Center.	Boolean	`False` (this value is the default)

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

sample astra_control_center.yaml

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  crds:
    externalTraefik: false
    externalCertManager: false
```

Complete Astra Control Center and operator installation

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

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

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

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

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

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

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

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

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

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

Verify system status

You can verify system status using kubectl commands. If you prefer to use OpenShift, you can use comparable oc commands for verification steps.

Steps

1. Verify that the installation process did not produce warnings messages related to the validation checks:

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

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

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

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

3. Verify that all system components installed successfully.

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

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

Expand for sample response

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

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

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

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

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

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

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

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

5. When all the pods are running, verify that the installation was successful (`READY` is `True`) and get the initial setup password you'll use when you log in to Astra Control Center:

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

Response:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  24.02.0-69
10.111.111.111  True
```

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

Set up ingress for load balancing

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

After Astra Control Center is deployed, you'll need to configure the ingress controller to expose Astra Control Center with a URL.

Setup steps differ depending on the type of ingress controller you use. Astra Control Center supports many ingress controller types. These setup procedures provide example steps for some common ingress controller types.

Before you begin

- The required ingress controller should already be deployed.
- The ingress class corresponding to the ingress controller should already be created.

Steps for Istio ingress

1. Configure Istio ingress.

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

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

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

   Sample command:

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

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

   Sample command:

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

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

4. Deploy an ingress resource in the `netapp-acc` (or custom-named) namespace using the v1 resource type for a schema (`istio-Ingress.yaml` is used in this example):

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

5. Apply the changes:

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

6. Check the status of the ingress:

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

Response:

```
NAME    CLASS HOSTS              ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

7. Finish Astra Control Center installation.

Steps for Nginx ingress controller

1. Create a secret of type `kubernetes.io/tls` for a TLS private key and certificate in `netapp-acc` (or custom-named) namespace as described in TLS secrets.
2. Deploy an ingress resource in `netapp-acc` (or custom-named) namespace using the v1 resource type for a schema (`nginx-Ingress.yaml` is used in this example):

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

3. Apply the changes:

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

NetApp recommends installing the nginx controller as a deployment rather than a `daemonSet`.

Steps for OpenShift ingress controller

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

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

Log in to the Astra Control Center UI

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

Steps

1. In a browser, enter the FQDN (including the `https://` prefix) you used in the `astraAddress` in the `astra_control_center.yaml` CR when you installed Astra Control Center.
2. Accept the self-signed certificates if prompted.

   You can create a custom certificate after login.

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

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

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

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

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

Troubleshoot the installation

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

Options

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

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

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

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

Alternative installation procedures

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

What's next

- (Optional) Depending on your environment, complete post-installation configuration steps.
- After you install Astra Control Center, log in to the UI, and change your password, you'll want to set up a license, add clusters, enable authentication, manage storage, and add buckets.

Configure an external cert manager

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

Steps

1. Confirm that you have a cert manager installed:

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

Sample response:

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

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

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

Sample response:

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

3. Create a secret with previously generated files:

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

Sample response:

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

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

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

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

Sample response:

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

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

```
kubectl get ClusterIssuer
```

Sample response:

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

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

Install Astra Control Center using OpenShift OperatorHub

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

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

Before you begin

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

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

- **Ensure healthy cluster operators and API services**:
  - From your OpenShift cluster, ensure all cluster operators are in a healthy state:

    ```
    oc get clusteroperators
    ```

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

    ```
    oc get apiservices
    ```

- **Ensure a routable FQDN**: The Astra FQDN you plan to use can be routed to the cluster. This means that you either have a DNS entry in your internal DNS server or you are using a core URL route that is already registered.
- **Obtain OpenShift permissions**: You'll need all necessary permissions and access to the Red Hat OpenShift Container Platform to perform the installation steps described.
- **Configure a cert manager**: If a cert manager already exists in the cluster, you need to perform some prerequisite steps so that Astra Control Center does not install its own cert manager. By default, Astra

Control Center installs its own cert manager during installation.

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

  a. Create the operator namespace:

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

  b. Complete setup for your ingress controller type.

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

You should also consider the following:

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

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

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

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

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

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

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

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

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

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

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

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

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

Steps

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

Do not delete the Astra Control Center operator (for example, `kubectl delete -f astra_control_center_operator_deploy.yaml`) at any time during Astra Control Center installation or operation to avoid deleting pods.

Download and extract Astra Control Center

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

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

Astra Control image registry

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

NetApp Support Site

1. Download the bundle containing Astra Control Center (`astra-control-center-[version].tar.gz`) from the Astra Control Center downloads page.
2. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (`astra-control-center-certs-[version].tar.gz`) to verify the signature of the bundle.

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

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

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

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

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

Complete additional steps if you use a local registry

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

Install the NetApp Astra kubectl plugin

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

Before you begin

NetApp provides plugin binaries for different CPU architectures and operating systems. You need to know which CPU and operating system you have before you perform this task.

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

Steps

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

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

```
ls kubectl-astra/
```

2. Move the correct binary into the current path and rename it to `kubectl-astra`:

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

Add the images to your registry

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

Docker

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

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

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

- Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.bundle.yaml`).
- Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, "https://<docker-registry>".
- Replace <MY_REGISTRY_USER> with the user name.
- Replace <MY_REGISTRY_TOKEN> with an authorized token for the registry.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

Podman

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

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

b. Log in to your registry:

```
podman login <YOUR_REGISTRY>
```

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

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

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

2. Change the directory:

```
cd manifests
```

Find the operator install page

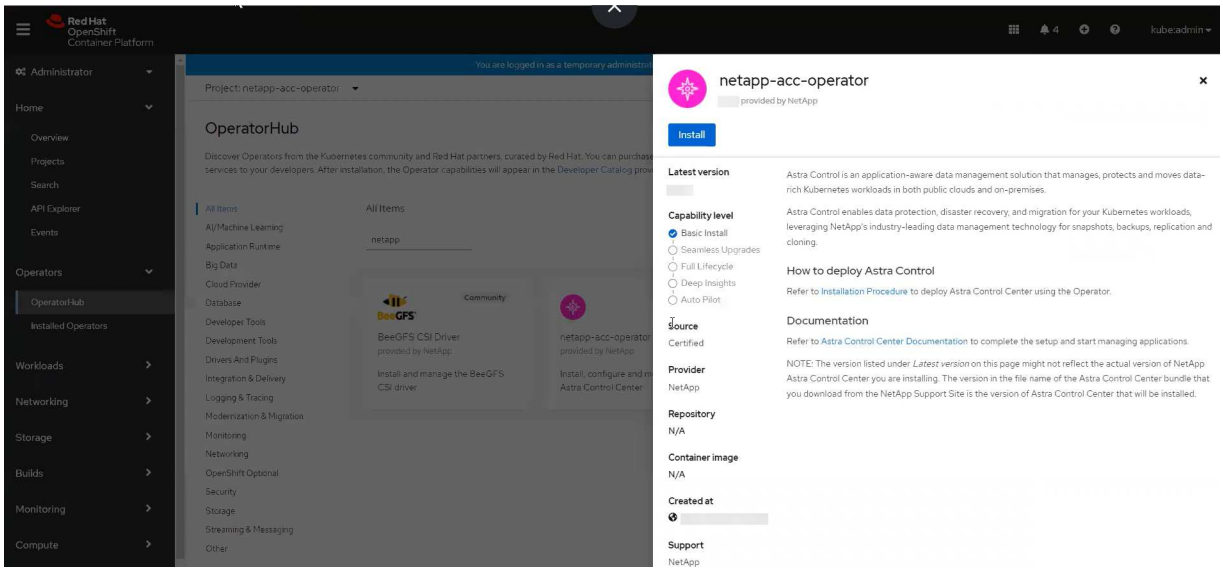
1. Complete one of the following procedures to access the operator install page:

Red Hat OpenShift web console

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

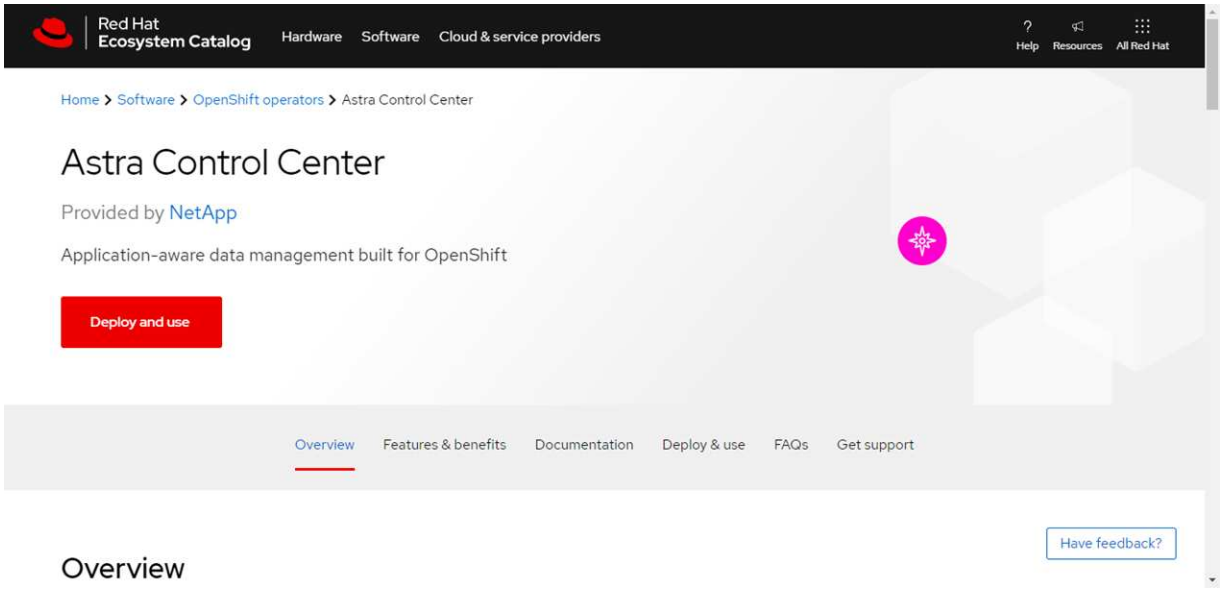
> You can upgrade only to the current version of Astra Control Center using this operator.

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



Red Hat Ecosystem Catalog

1. Select the NetApp Astra Control Center operator.
2. Select **Deploy and use**.



Install the operator

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

   The operator will be available in all cluster namespaces.

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

   b. Select a manual or automatic approval strategy.

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

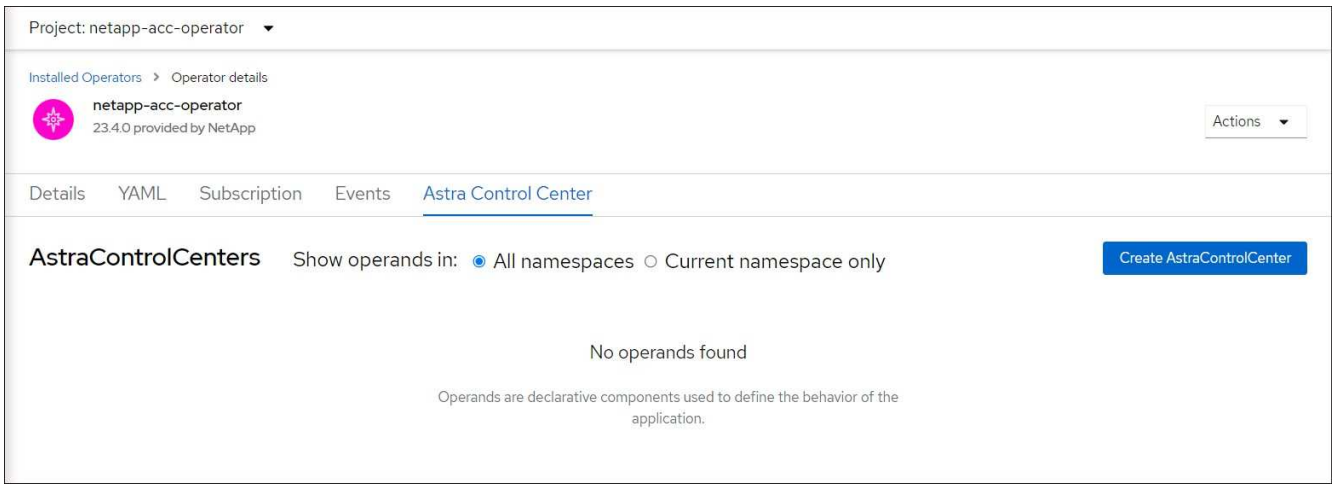
   c. Select **Install**.

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

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

Install Astra Control Center

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



2. Complete the `Create AstraControlCenter` form field:

   a. Keep or adjust the Astra Control Center name.

   b. Add labels for the Astra Control Center.

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

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

   e. Enter the Astra Control Center version; for example, 24.02.0-69.

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

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

   h. Select the scale size of the installation.

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

i. Select the ingress type:

- **Generic** (`ingressType: "Generic"`) (Default)

  Use this option when you have another ingress controller in use or would prefer to use your own ingress controller. After Astra Control Center is deployed, you'll need to configure the ingress controller to expose Astra Control Center with a URL.

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

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

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

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

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

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

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

l. Enter the admin first name.

m. Configure resources scaling.

n. Provide the default storage class.

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

o. Define CRD handling preferences.

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

Create a registry secret

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

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

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

2. Create a secret in this namespace:

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

Astra Control supports Docker registry secrets only.

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

What's next

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

Install Astra Control Center with a Cloud Volumes ONTAP storage backend

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

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

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

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

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

Deploy Astra Control Center in Amazon Web Services

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

What you'll need for AWS

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

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

Operational environment requirements for AWS

Astra Control Center requires the following operational environment for AWS:

- Red Hat OpenShift Container Platform 4.11 through 4.13

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

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

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

Overview of deployment for AWS

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

Each of these steps is explained in more detail below.

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

Ensure that you have sufficient IAM permissions

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

See Initial AWS credentials.

Install a RedHat OpenShift cluster on AWS

Install a RedHat OpenShift Container Platform cluster on AWS.

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

Configure AWS

Next, configure AWS to create a virtual network, set up EC2 compute instances, and create an AWS S3 bucket. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create an Elastic Container Registry (ECR) to host the Astra Control Center images, and push the images to this registry.

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

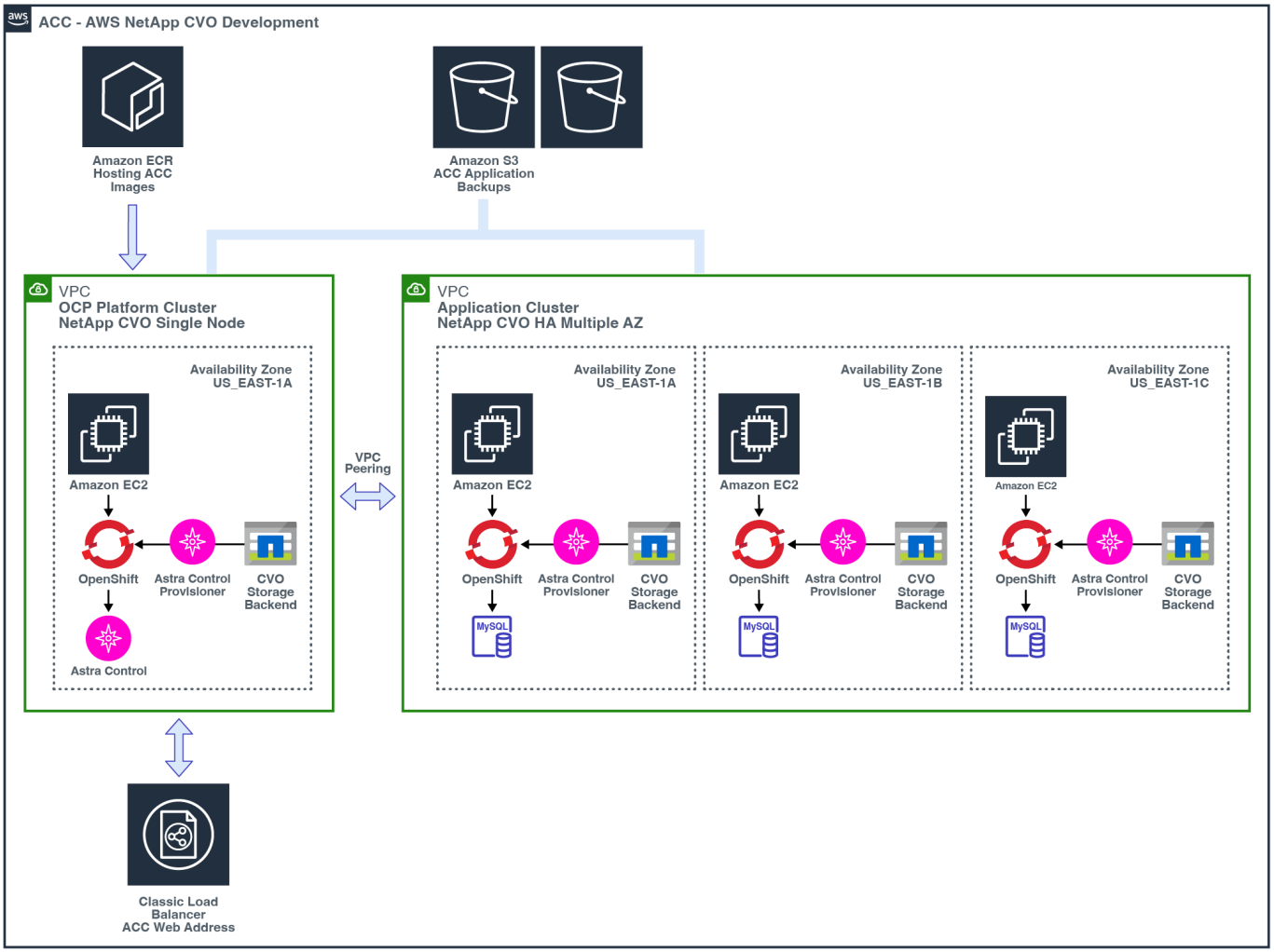
1. Create an AWS virtual network.
2. Review the EC2 compute instances. This can be a bare metal server or VMs in AWS.
3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in AWS to meet the Astra requirements. Refer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

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

   b. Push the Astra Control Center images to your defined registry.

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

Here's an example of an AWS deployment:



Configure NetApp BlueXP for AWS

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

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

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

Steps

1. Add your credentials to BlueXP.
2. Create a workspace.
3. Add a connector for AWS. Choose AWS as the Provider.
4. Create a working environment for your cloud environment.
    a. Location: "Amazon Web Services (AWS)"
    b. Type: "Cloud Volumes ONTAP HA"
5. Import the OpenShift cluster. The cluster will connect to the working environment you just created.
    a. View the NetApp cluster details by selecting **K8s** > **Cluster list** > **Cluster Details**.

b. In the upper right corner, note the Astra Control Provisioner version.

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

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

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

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

Install Astra Control Center for AWS

Follow the standard Astra Control Center installation instructions.

AWS uses the Generic S3 bucket type.

Deploy Astra Control Center in Google Cloud Platform

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

What you'll need for GCP

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

- Astra Control Center license. Refer to Astra Control Center licensing requirements.
- Meet Astra Control Center requirements.
- NetApp Cloud Central account
- If using OCP, Red Hat OpenShift Container Platform (OCP) 4.11 through 4.13
- If using OCP, Red Hat OpenShift Container Platform (OCP) permissions (on namespace level to create pods)
- GCP Service Account with permissions that enable you to create buckets and connectors

Operational environment requirements for GCP

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

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

Overview of deployment for GCP

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

Each of these steps is explained in more detail below.

1. Install a RedHat OpenShift cluster on GCP.

2. Create a GCP Project and Virtual Private Cloud.
3. Ensure that you have sufficient IAM permissions.
4. Configure GCP.
5. Configure NetApp BlueXP for GCP.
6. Install Astra Control Center for GCP.

Install a RedHat OpenShift cluster on GCP

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

For installation instructions, see the following:

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

Create a GCP Project and Virtual Private Cloud

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

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

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

Ensure that you have sufficient IAM permissions

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

See Initial GCP credentials and permissions.

Configure GCP

Next, configure GCP to create a VPC, set up compute instances, and create a Google Cloud Object Storage. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create a Google Container Registry to host the Astra Control Center images, and push the images to this registry.

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

1. Create a GCP Project and VPC in the GCP that you plan on using for the OCP cluster with CVO backend.
2. Review the compute instances. This can be a bare metal server or VMs in GCP.
3. If the instance type does not already match the Astra minimum resource requirements for master and worker nodes, change the instance type in GCP to meet the Astra requirements. Refer to Astra Control Center requirements.
4. Create at least one GCP Cloud Storage Bucket to store your backups.
5. Create a secret, which is required for bucket access.
6. (Optional) If you cannot access the NetApp image registry, do the following:
    a. Create a Google Container Registry to host the Astra Control Center images.
    b. Set up Google Container Registry access for Docker push/pull for all the Astra Control Center images.

Example: Astra Control Center images can be pushed to this registry by entering the following script:

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

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

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

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

7. Set up DNS zones.

Configure NetApp BlueXP for GCP

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

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

Before you begin

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

Steps

1. Add your credentials to BlueXP. See Adding GCP accounts.
2. Add a connector for GCP.
   a. Choose "GCP" as the Provider.
   b. Enter GCP credentials. See Creating a connector in GCP from BlueXP.
   c. Ensure that the connector is running and switch to that connector.
3. Create a working environment for your cloud environment.
   a. Location: "GCP"
   b. Type: "Cloud Volumes ONTAP HA"
4. Import the OpenShift cluster. The cluster will connect to the working environment you just created.

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

b. In the upper right corner, note the Astra Control Provisioner version.

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

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

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

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

Install Astra Control Center for GCP

Follow the standard Astra Control Center installation instructions.

GCP uses the Generic S3 bucket type.

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

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

Deploy Astra Control Center in Microsoft Azure

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

What you'll need for Azure

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

- Astra Control Center license. Refer to Astra Control Center licensing requirements.
- Meet Astra Control Center requirements.
- NetApp Cloud Central account
- If using OCP, Red Hat OpenShift Container Platform (OCP) 4.11 through 4.13
- If using OCP, Red Hat OpenShift Container Platform (OCP) permissions (on namespace level to create pods)
- Azure credentials with permissions that enable you to create buckets and connectors

Operational environment requirements for Azure

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

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

to Astra Control Center operational environment requirements.

Overview of deployment for Azure

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

Each of these steps is explained in more detail below.

1. Install a RedHat OpenShift cluster on Azure.
2. Create Azure resource groups.
3. Ensure that you have sufficient IAM permissions.
4. Configure Azure.
5. Configure NetApp BlueXP (formerly Cloud Manager) for Azure.
6. Install and configure Astra Control Center for Azure.

Install a RedHat OpenShift cluster on Azure

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

For installation instructions, see the following:

- Installing OpenShift cluster on Azure.
- Installing an Azure account.

Create Azure resource groups

Create at least one Azure resource group.

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

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

Ensure that you have sufficient IAM permissions

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

See Azure credentials and permissions.

Configure Azure

Next, configure Azure to create a virtual network, set up compute instances, and create an Azure Blob container. If you cannot access the NetApp Astra Control Center image registry, you'll also need to create an Azure Container Registry (ACR) to host the Astra Control Center images, and push the images to this registry.

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

1. Create an Azure virtual network.
2. Review the compute instances. This can be a bare metal server or VMs in Azure.
3. If the instance type does not already match the Astra minimum resource requirements for master and

worker nodes, change the instance type in Azure to meet the Astra requirements. Refer to Astra Control Center requirements.

4. Create at least one Azure Blob container to store your backups.
5. Create a storage account. You'll need a storage account to create a container to be used as a bucket in Astra Control Center.
6. Create a secret, which is required for bucket access.
7. (Optional) If you cannot access the NetApp image registry, do the following:
   a. Create an Azure Container Registry (ACR) to host the Astra Control Center images.
   b. Set up ACR access for Docker push/pull for all the Astra Control Center images.
   c. Push the Astra Control Center images to this registry using the following script:

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

   Example:

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

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. Set up DNS zones.

Configure NetApp BlueXP (formerly Cloud Manager) for Azure

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

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

Before you begin

Access to the Azure account with the required IAM permissions and roles

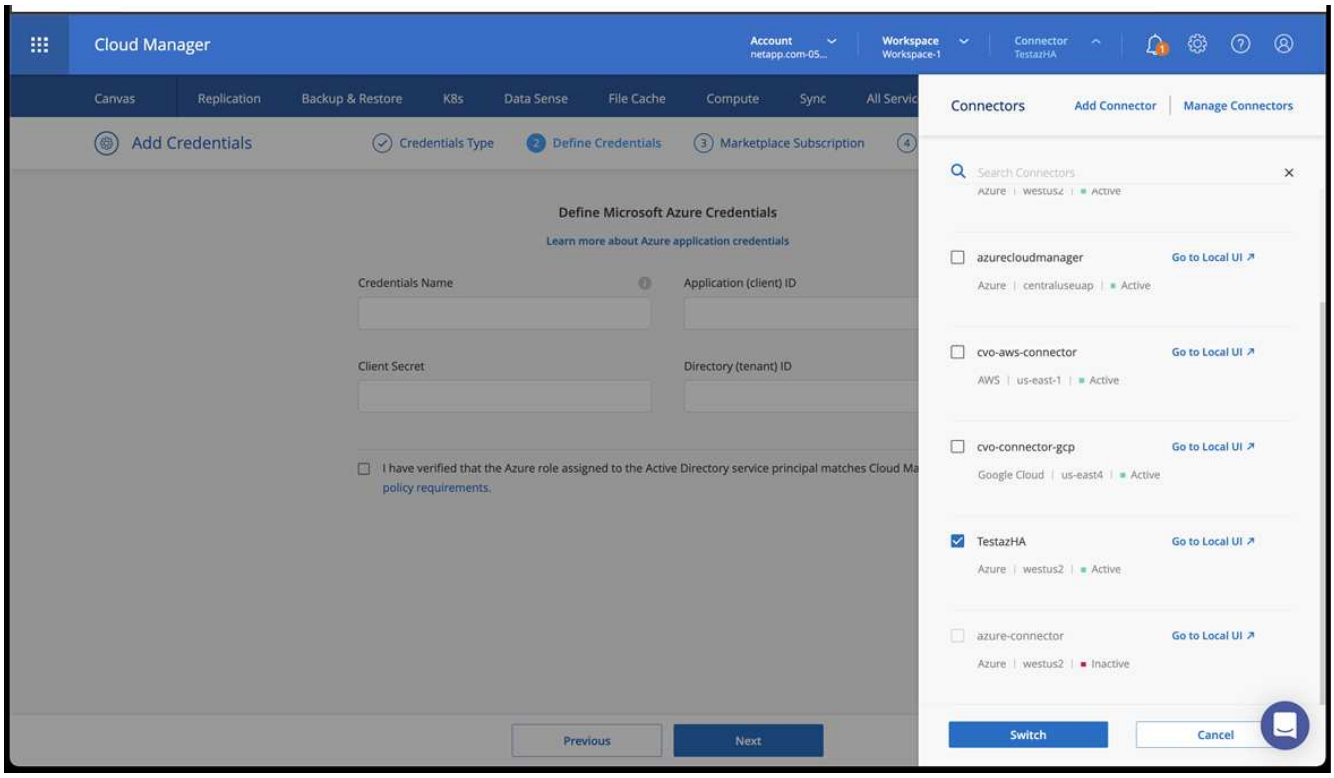
Steps

1. Add your credentials to BlueXP.
2. Add a connector for Azure. See BlueXP policies.

a. Choose **Azure** as the Provider.
b. Enter Azure credentials, including the application ID, client secret, and directory (tenant) ID.

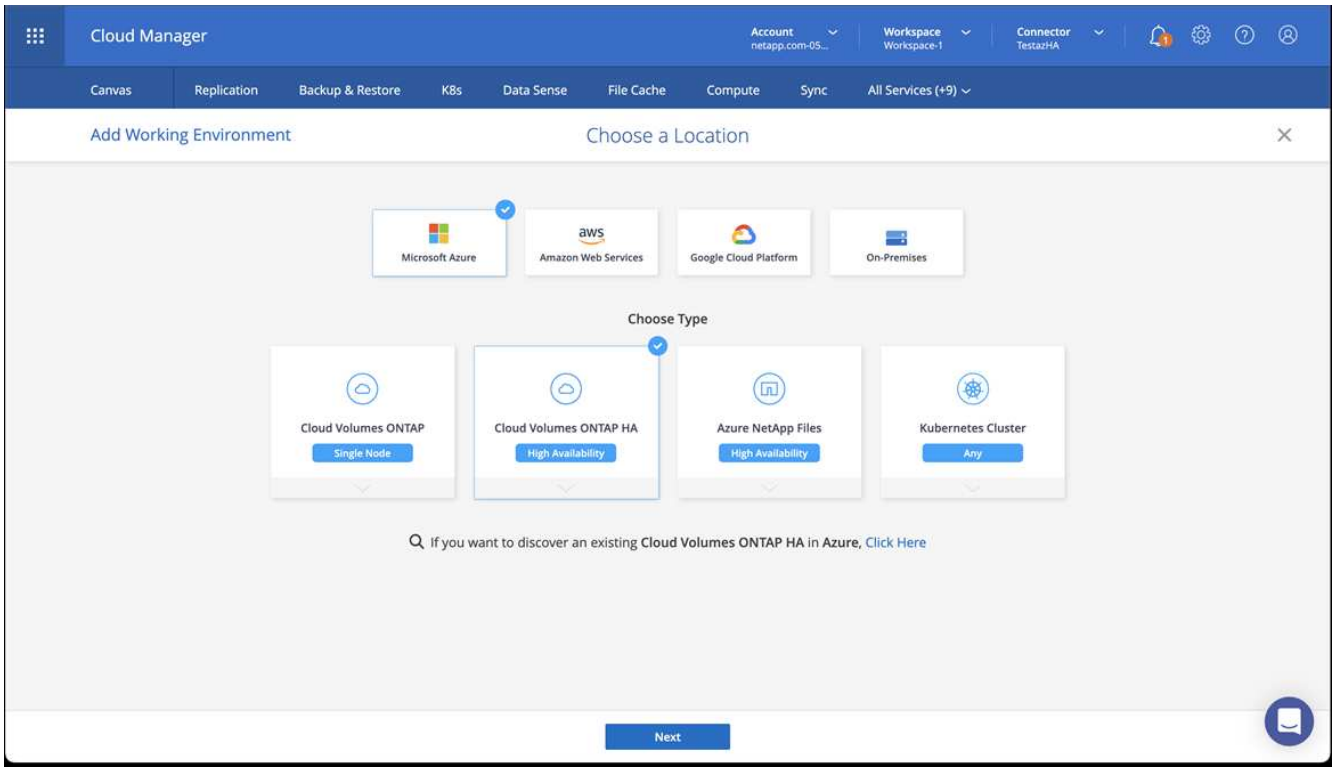
   See Creating a connector in Azure from BlueXPr.

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



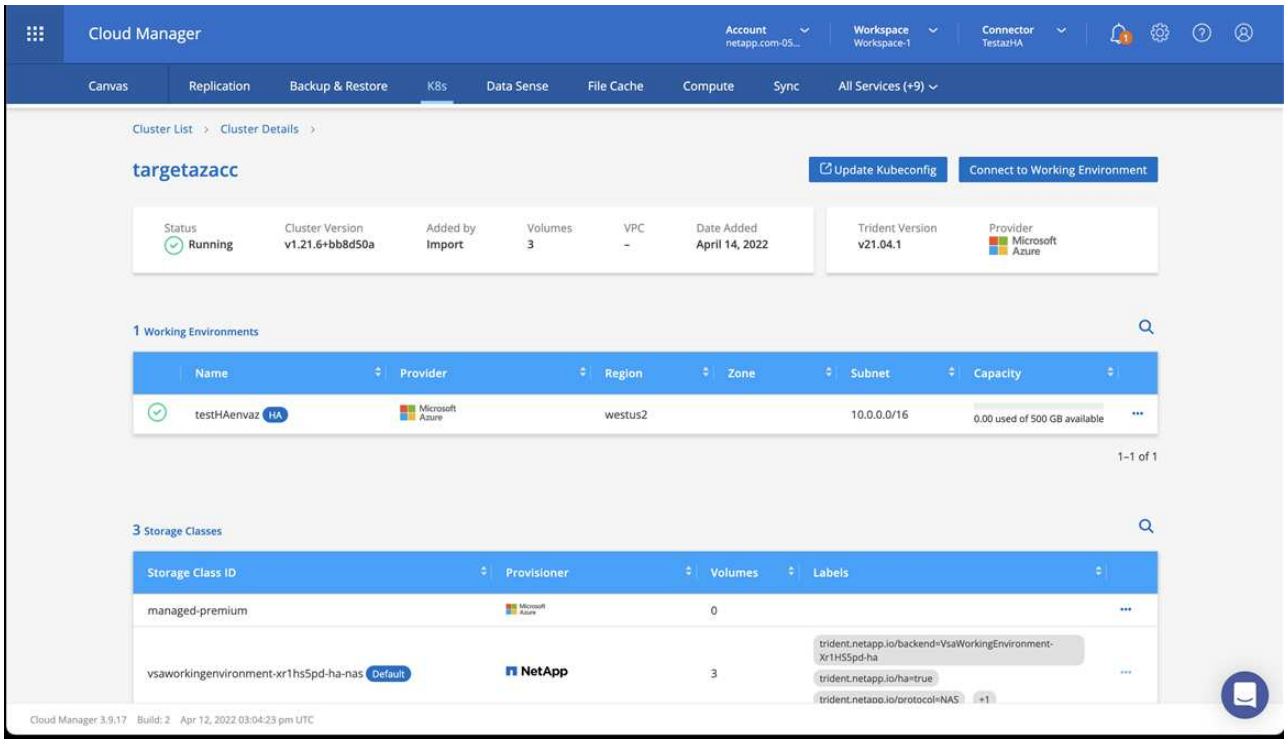
4. Create a working environment for your cloud environment.

   a. Location: "Microsoft Azure".
   b. Type: "Cloud Volumes ONTAP HA".



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

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



   b. In the upper right corner, note the Astra Control Provisioner version.

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

This imports your Red Hat OpenShift cluster and assigns a default storage class. You select the storage class.

Astra Control Provisioner is automatically installed as part of the import and discovery process.

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

Install and configure Astra Control Center for Azure

Install Astra Control Center with the standard installation instructions.

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

Configure Astra Control Center after installation

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

Remove resource limitations

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

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

Steps

1. Get the resource quotas in the `netapp-acc` (or custom-named) namespace:

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

Response:

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

2. Delete all of the resource quotas by name:

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

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

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

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

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

Response:

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

4. Delete the limit ranges by name:

```
kubectl delete limitrange cpu-limit-range -n [netapp-acc or custom
namespace]
```

Add a custom TLS certificate

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

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

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

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

Before you begin

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

Remove the self-signed certificate

Remove the existing self-signed TLS certificate.

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

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

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

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

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

Add a new certificate using the command line

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

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

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

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

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

CRD:

```
#spec:
#  selfSigned: {}

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

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

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

Response:

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

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

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

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

5. Create the certificate using the following command:

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

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

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

Response:

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

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

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

CRD:

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

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

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

CRD:

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

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

Set up Astra Control Center

Add a license for Astra Control Center

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

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

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

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

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

Before you begin

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

Steps

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

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

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

Enable Astra Control Provisioner

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

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

About this task

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

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

Before you begin

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

Astra Control Provisioners users with Astra Control Center

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

Astra Control Provisioner only users

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

(Step 1) Get the Astra Control Provisioner image

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

Astra Control image registry

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

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

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

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

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

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

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

   Example:

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

   b. Tag the image:

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

   c. Push the image to your custom registry:

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

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

NetApp Support Site

1. Download the Astra Control Provisioner bundle (`trident-acp-[version].tar`) from the Astra Control Center downloads page.
2. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (astra-control-center-certs-[version].tar.gz) to verify the signature of the trident-acp-[version] tar bundle.

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

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

3. Load the Astra Control Provisioner image:

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

   Response:

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

4. Tag the image:

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

5. Push the image to your custom registry:

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

(Step 2) Enable Astra Control Provisioner in Astra Trident

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

Astra Trident operator

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

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

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

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

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

4. Verify Astra Trident is running:

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

   Response:

```
NAME      AGE
trident   21m
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Response:

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

tridentctl

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

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

4. Confirm that Astra Control Provisioner has been enabled:

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

Response:

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

Helm

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

```
kubectl delete psp tridentoperatorpod
```

3. Add the Astra Trident Helm repository:

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

4. Update the Helm chart:

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

Response:

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

5. List the images:

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

Response:

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

6. Ensure that trident-operator 24.02.0 is available:

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

Response:

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

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

Options

- Images without a registry

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

- Images in one or more registries

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

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

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

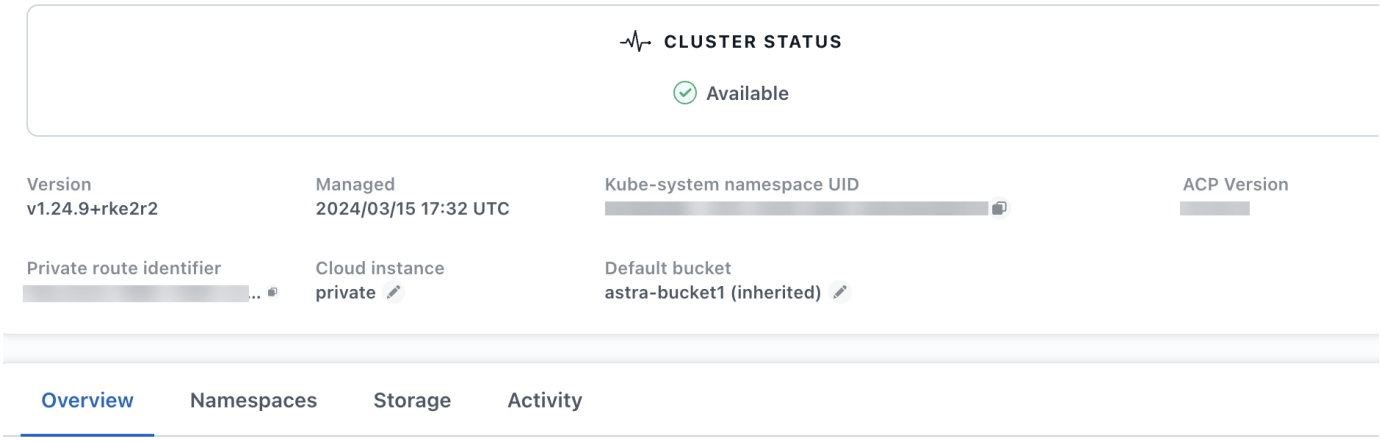
```
./tridentctl uninstall -n trident
```

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

Result

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

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



For more information

- Astra Trident upgrades documentation

Prepare your environment for cluster management using Astra Control

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

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

Before you begin

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

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

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

  - `netapp monitoring` namespace:

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

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

  Run the following commands in the ONTAP command line:

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

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

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

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

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

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

Run eligibility checks

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

Steps

1. Determine the Astra Trident version you are running:

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

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

   ```
   NAME      VERSION
   trident   24.02.0
   ```

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

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

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

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

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

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

```
kubectl get sc
```

Sample response:

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

Create a cluster role kubeconfig

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

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

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

Before you begin

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

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

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

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

Steps

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

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

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

b. Apply the service account:

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

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

Limited cluster role

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

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

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

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

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

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

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

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

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

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

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

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

3. Apply the cluster role:

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

Expanded cluster role

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

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

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

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

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

2. Apply the cluster role:

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

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

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

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

   b. Apply the cluster role binding:

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

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

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

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

   b. Apply the token secret:

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

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

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

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

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

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

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

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

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

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

7. Generate the kubeconfig as follows:
   a. Create a `create-kubeconfig.sh` file.
   b. Replace `TOKEN_INDEX` in the beginning of the following script with the correct value.

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

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

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

CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

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

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
```

```
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token-
user

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

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

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

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

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

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

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(Tech preview) Install Astra Connector for managed clusters

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

Install Astra Connector

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

About this task

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

Before you begin

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

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

Steps

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

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

2. Verify that the operator is installed and ready:

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

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

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

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

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

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

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

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

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

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

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

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

8. Verify that the Astra Connector is fully deployed:

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

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

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

You should see output similar to the following:

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

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

Add a cluster

To begin managing your apps, add a Kubernetes cluster and manage it as a compute resource. You have to add a cluster for Astra Control Center to discover your Kubernetes applications.

We recommend that Astra Control Center manage the cluster it is deployed on first before you add other clusters to Astra Control Center to manage. Having the initial cluster under management is necessary to send Kubemetrics data and cluster-associated data for metrics and troubleshooting.

Before you begin

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

Steps

1. Navigate from either the Dashboard or the Clusters menu:
   - From **Dashboard** in the Resource Summary, select **Add** from the Clusters pane.
   - In the left navigation area, select **Clusters** and then select **Add Cluster** from the Clusters page.
2. In the **Add Cluster** window that opens, upload a `kubeconfig.yaml` file or paste the contents of a `kubeconfig.yaml` file.

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

   If you create your own `kubeconfig` file, you should define only **one** context element in it. Refer to Kubernetes documentation for information about creating `kubeconfig` files. If you created a kubeconfig for a limited cluster role using this process, be sure to upload or paste that kubeconfig in this step.

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

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

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

Result

The cluster enters **Discovering** state and then changes to **Healthy**. You are now managing the cluster with Astra Control Center.

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

Enable authentication on an ONTAP storage backend

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

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

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

Enable credential-based authentication

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

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

A sample backend definition looks like this:

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

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

be performed by the Kubernetes or storage administrator.

Enable certificate-based authentication

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

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

You can use one of the following types of certificates:

- Self-signed certificate
- Third-party certificate

Enable authentication with a self-signed certificate

A typical workflow involves the following steps.

Steps

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

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

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

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

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

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

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

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

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

Enable authentication with a third-party certificate

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

Steps

1. Generate the private key and CSR:

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

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

   You now have the following three files:

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

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

Expand for sample.yaml

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

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

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


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

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

CA:
serial:

The certificate's generated name for reference:


===

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

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

Please enter Private Key: Press <Enter> when done

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

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

```
certificates {y|n}: n

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

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


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

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

==
# Verify if the certificate works fine:

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

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

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

Expand for sample.yaml

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

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

Please enter Certificate: Press <Enter> when done

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

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

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


==

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

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

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

==

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

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

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


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

Add a storage backend

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

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

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

Steps

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

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

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

Result

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

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

Add a bucket

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

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

Before you begin

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

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

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

Steps

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

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

4. Enter an existing bucket name and optional description.

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

5. Enter the name or IP address of the S3 endpoint.
6. Under **Select Credentials**, choose either the **Add** or **Use existing** tab.
   - If you chose **Add**:
     a. Enter a name for the credential that distinguishes it from other credentials in Astra Control.
     b. Enter the access ID and secret key by pasting the contents from your clipboard.
   - If you chose **Use existing**:
     a. Select the existing credentials you want to use with the bucket.
7. Select `Add`.

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

Concepts

Architecture and components

Astra Control is a Kubernetes application data lifecycle management solution that simplifies operations for stateful applications, and helps you store, protect, and move your Kubernetes workloads across hybrid and multi-cloud environments.

Capabilities

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

Store:

- Dynamic storage provisioning for containerized workloads
- In-flight encryption of data from container to persistent volumes
- Cross-region, cross-zone replication

Protect:

- Automated discovery and application-aware protection of an entire application and its data
- Instant recovery of an application from any snapshot version based on your organization’s needs
- Fast failover across zones, regions and cloud providers

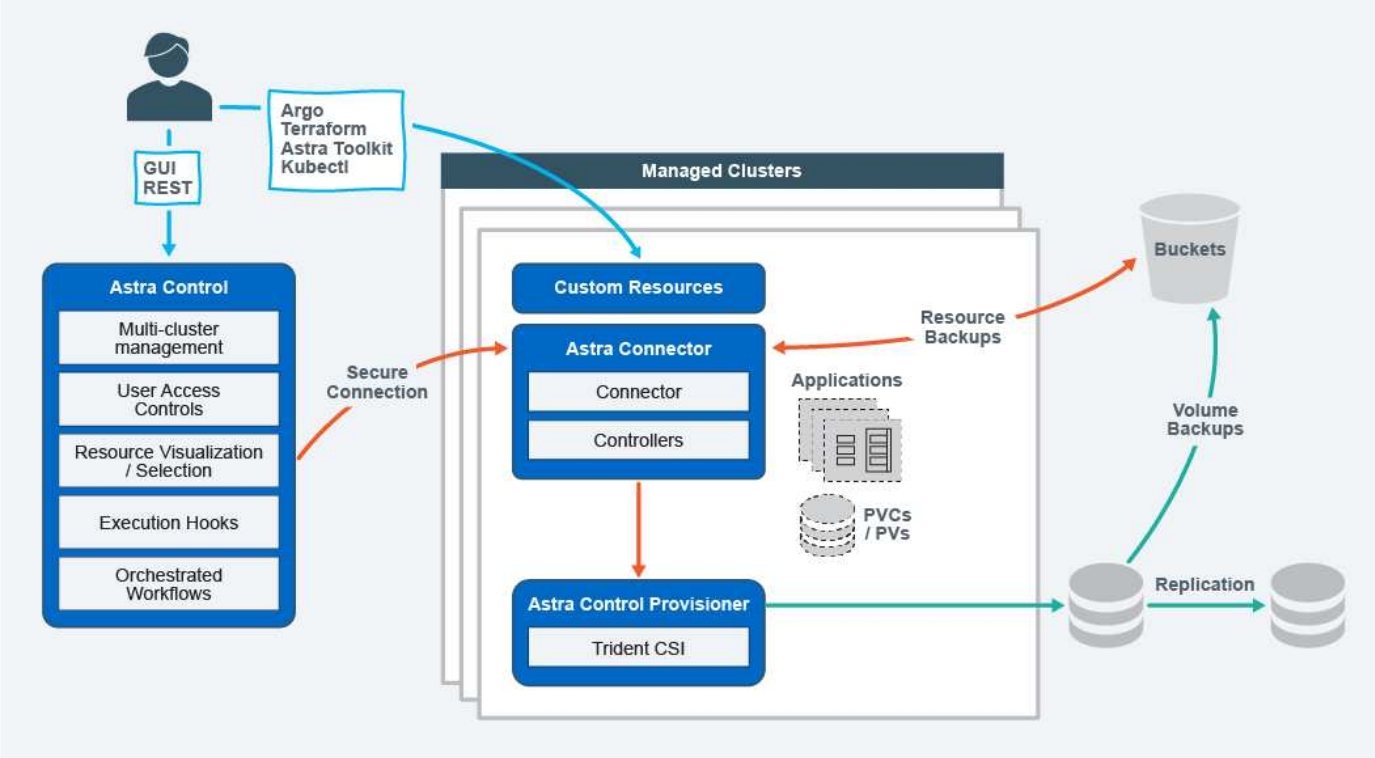
Move:

- Complete application and data mobility within and between Kubernetes clusters and clouds
- Instant clones of entire applications and data
- One click migration of applications through consistent web UI and API

Architecture

The architecture of Astra Control enables it to provide advanced data management capabilities that enhance both the functionality and availability of Kubernetes applications, simplifies the management, protection, and movement of containerized workloads across public clouds and on-prem environments, and provides automation capabilities through its REST API and SDK, enabling programmatic access for seamless integration with existing workflows.

Astra Control is Kubernetes-native, enabling data protection workflows that utilize custom resources while staying backward-compatible with the existing API and SDK. Kubernetes-native data protection offers significant advantages; by seamlessly integrating with Kubernetes APIs and resources, data protection can become an inherent part of the application lifecycle through an organization’s existing CI/CD and/or GitOps tools.



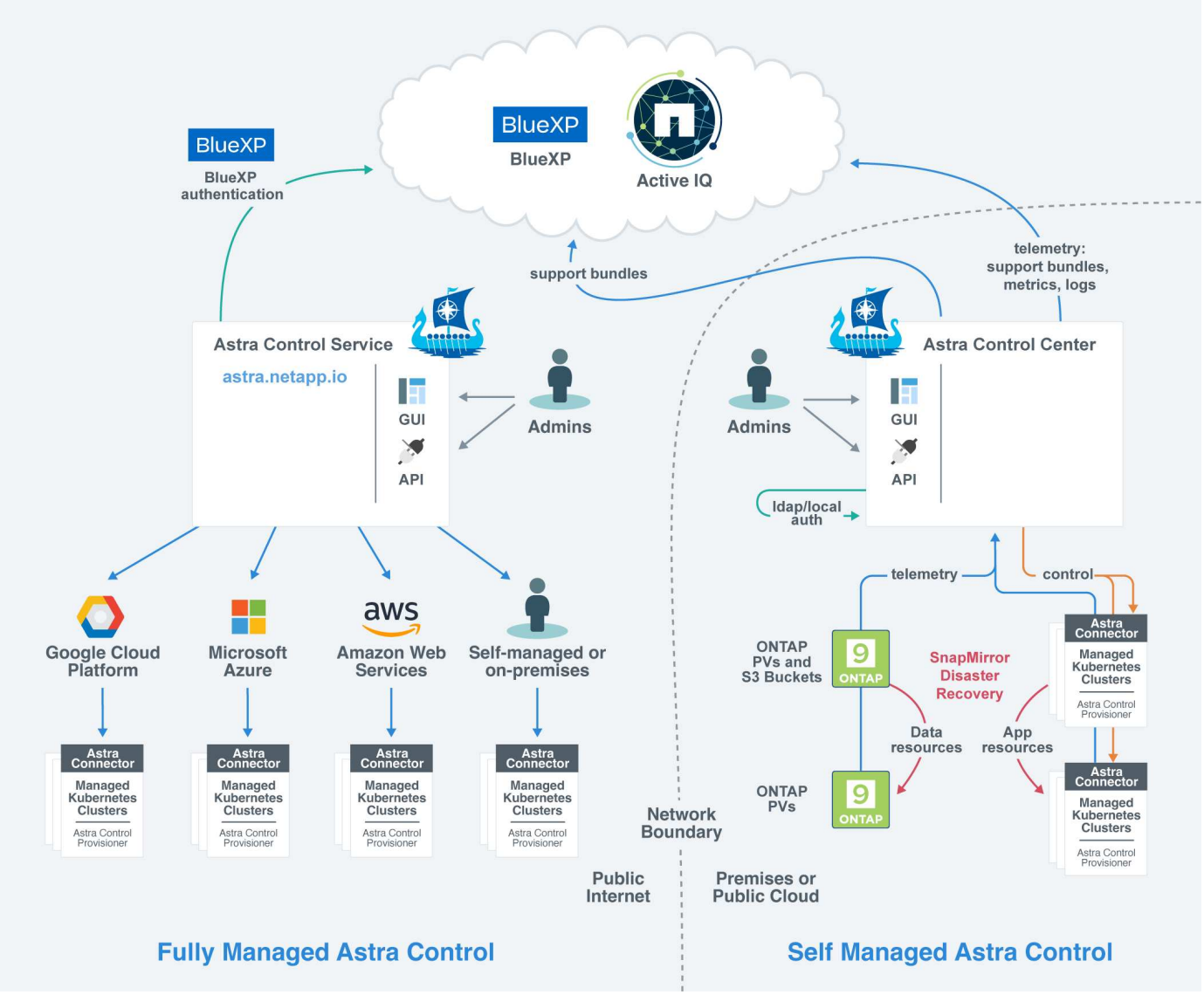
Astra Control is built on four complementary components:

- **Astra Control**: Astra Control is the centralized management service for all managed clusters, providing orchestrated workloads for application protection and mobility in the cloud and on-premises as well as the following capabilities:
  - Combined view of multiple clusters and clouds
  - Protection of orchestrated workflows
  - Granular resource visualization and selection
- **Astra Connector**: Astra Connector teams with Astra Control to provide a secure connection to each managed cluster, offering local execution of scheduled operations regardless of connection status as well as the following capabilities:
  - Local execution of scheduled operations regardless of connection status
  - Local operations that distribute and optimize system resource usage of Astra across clusters
  - Local installation that enables least privilege access to the cluster for improved security
- **Astra Control Provisioner**: Astra Control Provisioner delivers core CSI provisioning functionality and advanced storage management capabilities for added security and disaster recovery configuration, as well as the following capabilities:
  - Dynamic storage provisioning for containerized workloads
  - Advanced storage management:
    - In-flight encryption of data from container to PV
    - SnapMirror Cloud functionality with cross-region, cross-zone replication
- **Astra Custom resources**: Custom resources used on each cluster provide a Kubernetes-native approach to running operations locally, simplifying integration with other Kubernetes-friendly tooling and automation as well as providing the following capabilities:
  - Direct ecosystem tool integration and automation workflows

- Lower-level primitives that enable custom workflows

Deployment models

Astra Control is available in two deployment models.



- **Astra Control Service**: A NetApp-managed service that provides application-aware data management of Kubernetes clusters in multiple cloud provider environments as well as self-managed Kubernetes clusters.

  Astra Control Service documentation

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

  Astra Control Center documentation

	Astra Control Service	Astra Control Center
How is it offered?	As a fully managed cloud service from NetApp	As software that you can download, install, and manage
Where is it hosted?	On a public cloud of NetApp's choice	On your own Kubernetes cluster
How is it updated?	Managed by NetApp	You manage any updates
What are the supported Kubernetes distributions?	• **Cloud providers** ◦ Amazon Web Services ▪ Amazon Elastic Kubernetes Service (EKS) ◦ Google Cloud ▪ Google Kubernetes Engine (GKE) ◦ Microsoft Azure ▪ Azure Kubernetes Service (AKS) • **Self-managed clusters** ◦ Kubernetes (Upstream) ◦ Rancher Kubernetes Engine (RKE) ◦ Red Hat OpenShift Container Platform • **On-premises clusters** ◦ Red Hat OpenShift Container Platform on-premises	• Azure Kubernetes Service on Azure Stack HCI • Google Anthos • Kubernetes (Upstream) • Rancher Kubernetes Engine (RKE) • Red Hat OpenShift Container Platform

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

For more information

- Astra Control Service documentation
- Astra Control Center documentation
- Astra Trident documentation
- 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

Both snapshots and backups protect the following types of data:

- The application itself
- Any persistent data volumes associated with the application
- Any resource artifacts belonging to the application

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. Snapshots are useful for cloning or restoring an app within the same cluster.

A *backup* is based on a snapshot. It is stored in the external object store, and because of this, 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, and set different schedule granularity levels. 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 and service level agreement (SLA) requirements.

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.

Immutable backups

An immutable backup is a backup that cannot be changed or deleted during a specified period. When you create an immutable backup, Astra Control checks to ensure that the bucket you are using is a write once read many (WORM) bucket, and if so, ensures that the backup is immutable from within Astra Control.
Astra Control Center supports creating immutable backups with the following platforms and bucket types:

- Amazon Web Services using an Amazon S3 bucket with S3 Object Lock configured
- NetApp StorageGRID using an S3 bucket with S3 Object Lock configured

Note the following when working with immutable backups:

- If you back up to a WORM bucket in an unsupported platform or to an unsupported bucket type, you might get unpredictable results, such as backup deletion failing even if the retention time has elapsed.

- Astra Control does not support data lifecycle management policies or manual deletion of objects on the buckets you use with immutable backups. Make sure that your storage backend is not configured to manage the lifecycle of Astra Control snapshots or backed up data.

Clones

A *clone* is an exact duplicate of an app, its configuration, and its persistent data volumes. 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.

Replication between storage backends

Using Astra Control, you can build business continuity for your applications with a low-RPO (Recovery Point Objective) and low-RTO (Recovery Time Objective) using asynchronous replication capabilities of NetApp SnapMirror technology. Once configured, this enables your applications to replicate data and application changes from one storage backend to another, on the same cluster or between different clusters.

You can replicate between two ONTAP SVMs on the same ONTAP cluster or on different ONTAP clusters.

Astra Control asynchronously replicates app snapshot copies to a destination cluster. The replication process includes data in the persistent volumes replicated by SnapMirror and the app metadata protected by Astra Control.

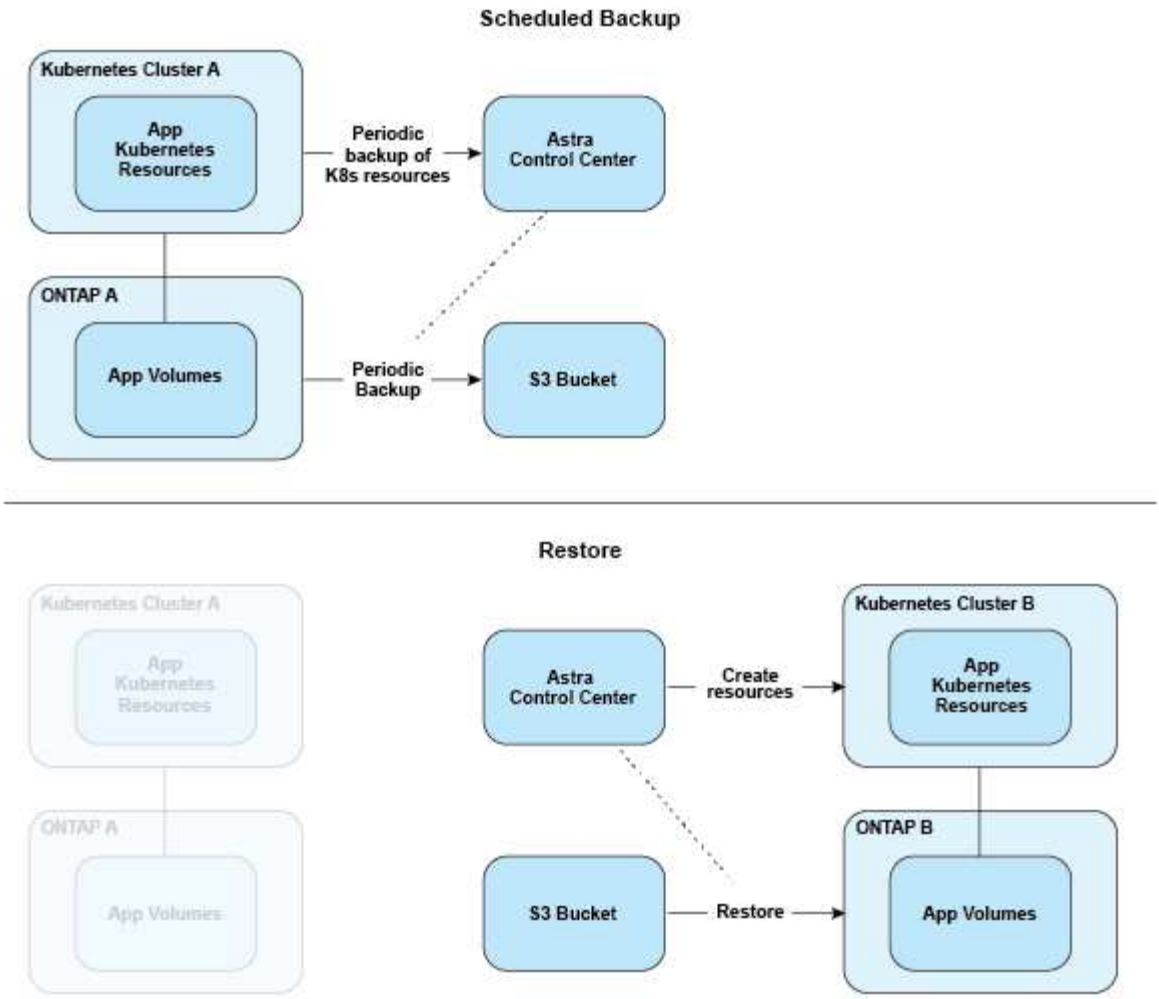
App replication is different from app backup and restore in the following ways:

- **App replication**: Astra Control requires the source and destination Kubernetes clusters (which can be the same cluster) to be available and managed with their respective ONTAP storage backends configured to enable NetApp SnapMirror. Astra Control takes the policy-driven application snapshot and replicates it to the destination storage backend. NetApp SnapMirror technology is used to replicate the persistent volume data. To fail over, Astra Control can bring the replicated app online by recreating the app objects on the destination Kubernetes cluster with the replicated volumes on the destination ONTAP cluster. Because the persistent volume data is already present on the destination ONTAP cluster, Astra Control can offer quick recovery times for failover.
- **App backup and restore**: When backing up applications, Astra Control creates a snapshot of the app data and stores it in an object storage bucket. When a restore is needed, the data in the bucket must be copied to a persistent volume on the ONTAP cluster. The backup/restore operation does not require the secondary Kubernetes/ONTAP cluster to be available and managed, but the additional data copy can result in longer restore times.

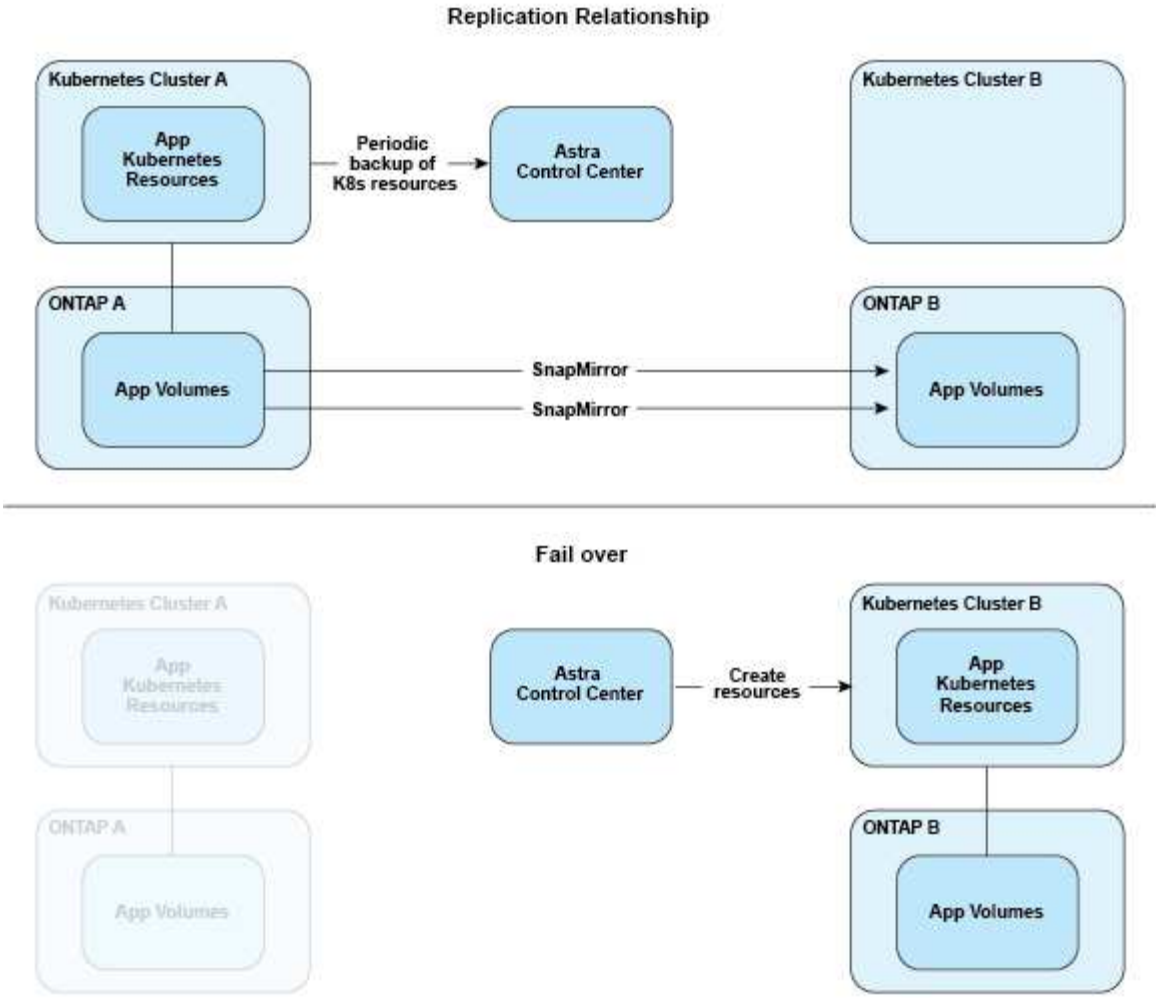
To learn how to replicate apps, refer to Replicate apps to a remote system using SnapMirror technology.

The following images show the scheduled backup and restore process compared to the replication process.

The backup process copies data to S3 buckets and restores from S3 buckets:



On the other hand, replication is done by replicating to ONTAP and then a failover creates the Kubernetes resources:



Backups, snapshots, and clones with an expired license

If your license expires, you can add a new application or perform application protection operations (such as snapshots, backups, clones, and restore operations) only if the application you are adding or protecting is another Astra Control Center instance.

Licensing

When you deploy Astra Control Center, it is installed with an embedded 90-day evaluation license for 4,800 CPU units. If you need more capacity or a longer evaluation period, or want to upgrade to a full license, you can obtain a different evaluation license or full license from NetApp.

You obtain a license in one of the following ways:

- If you are evaluating Astra Control Center and need different evaluation terms than what is included in the embedded evaluation license, contact NetApp to request a different evaluation license file.
- If you already purchased Astra Control Center, generate your NetApp license file (NLF) by logging in to the NetApp Support Site and navigating to your software licenses under the Systems menu.

For details about licenses needed for ONTAP storage backends, refer to supported storage backends.

Make sure that your license enables at least as many CPU units as you need. If the number of CPU units that Astra Control Center is currently managing exceeds the available CPU units in the new license being applied, you'll not be able to apply the new license.

Evaluation licenses and full licenses

An embedded evaluation license is provided with a new Astra Control Center installation. An evaluation license enables the same capabilities and features as a full license for a limited (90 day) period. After the evaluation period, a full license is required to continue with full functionality.

License expiration

If the active Astra Control Center license expires, UI and API functionality for the following features are unavailable:

- Manual local snapshots and backups
- Scheduled local snapshots and backups
- Restoring from a snapshot or backup
- Cloning from a snapshot or current state
- Managing new applications
- Configuring replication policies

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.

When you start managing an app on a cluster, all of that cluster's CPU units are included in the Astra Control Center license consumption, except Red Hat OpenShift cluster node CPU units that are reported by a using the label `node-role.kubernetes.io/infra: ""`.

Red Hat OpenShift infrastructure nodes do not consume licenses in Astra Control Center. To mark a node as an infrastructure node, apply the label `node-role.kubernetes.io/infra: ""` to the node.

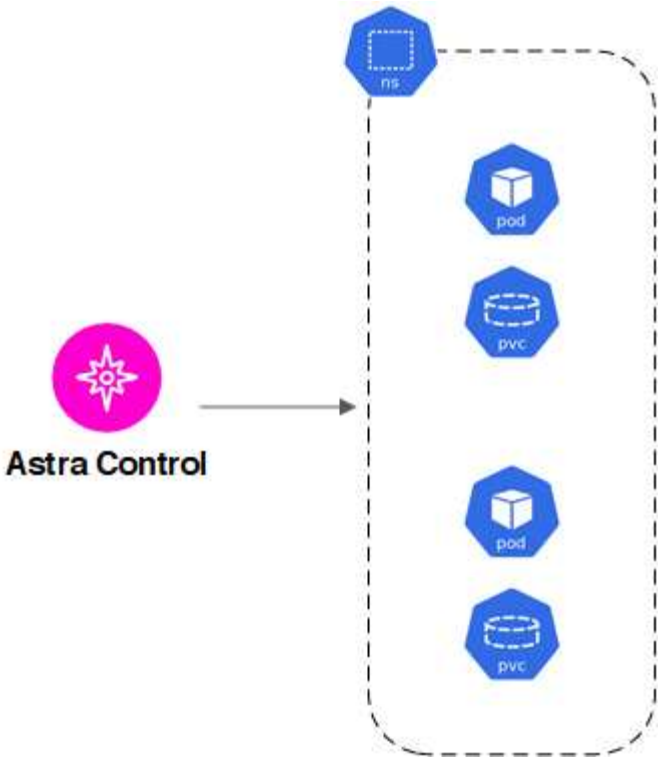
Find more information

- Add a license when you first set up Astra Control Center
- Update an existing license

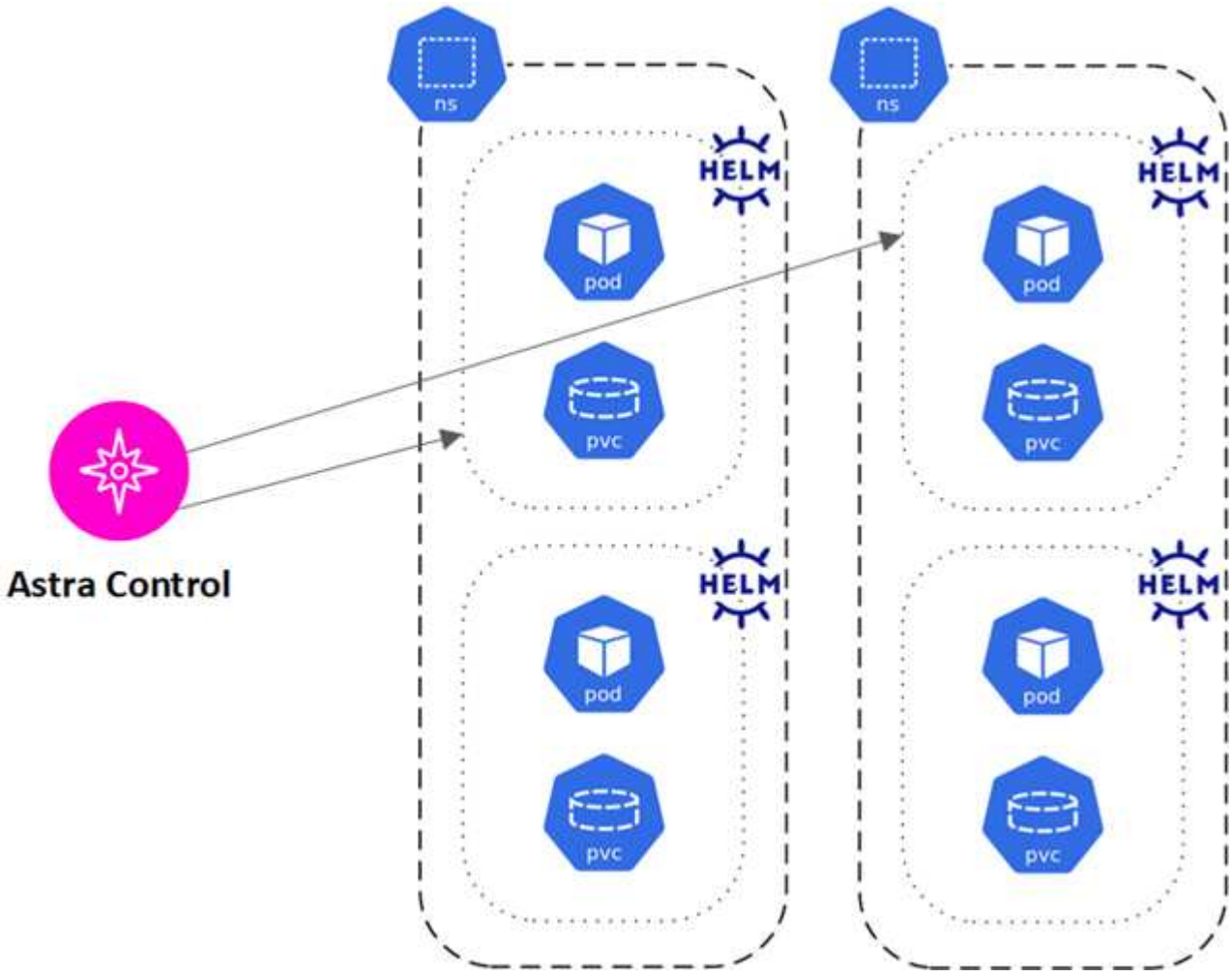
App management

When Astra Control discovers your clusters, the apps on those clusters are unmanaged until you choose how you want to manage them. A managed application in Astra Control can be any of the following:

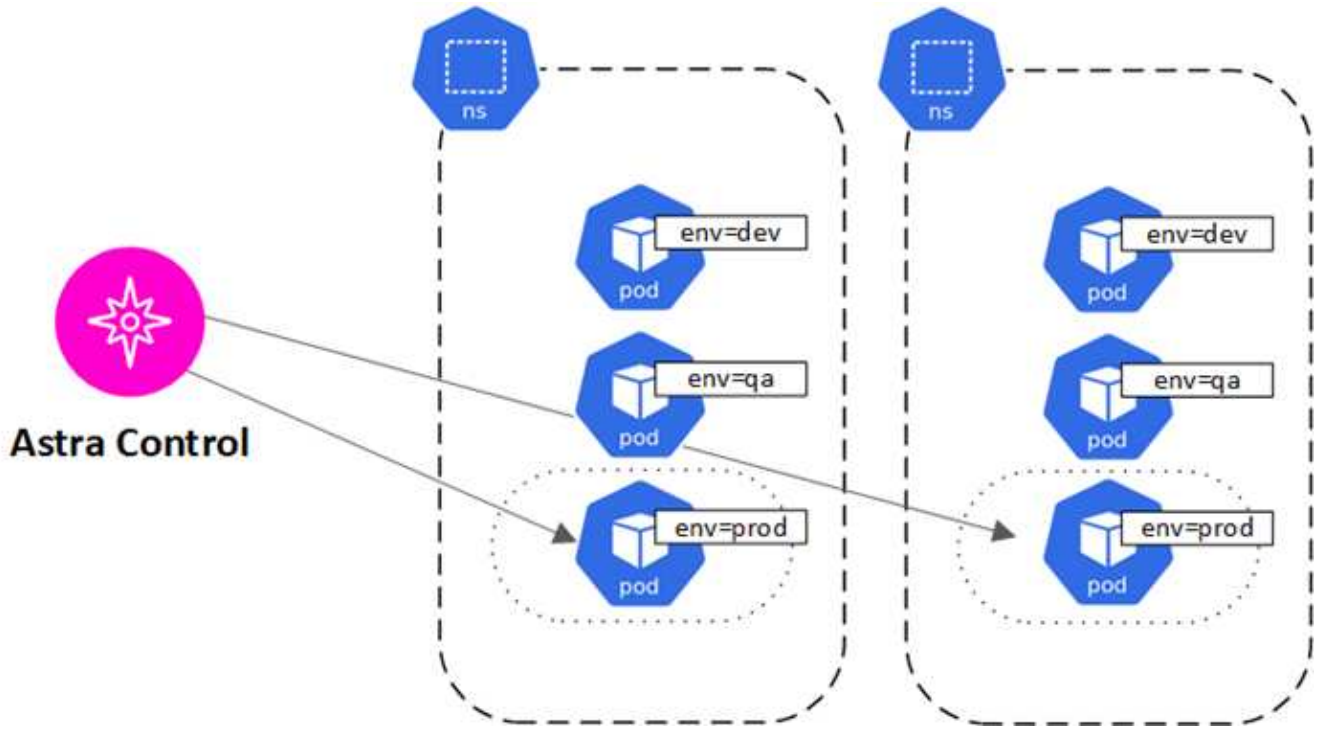
- A namespace, including all resources in that namespace



- An individual application deployed within one or more namespaces (helm3 is used in this example)



- A group of resources that are identified by a Kubernetes label within one or more namespaces



Storage classes and persistent volume size

Astra Control Center supports NetApp ONTAP and Longhorn as storage backends.

Overview

Astra Control Center supports the following:

- **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 monitoring information.
- **CSI-based storage classes backed by Longhorn**: You can use Longhorn with the Longhorn Container Storage Interface (CSI) driver.

Storage classes should be configured using Astra Control Provisioner.

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.

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 local users and roles

Use Astra Control Center

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 Applications page in Astra Control to define the apps and their resources.

You can define and manage apps that include storage resources with running pods, or apps that include storage resources without any running pods. Apps that have no running pods are known as data-only applications.

Application management requirements

Astra Control has the following application management requirements:

- **Licensing**: To manage applications using Astra Control Center, you need either the embedded Astra Control Center evaluation license or a full license.
- **Namespaces**: Apps can be defined within one or more specified namespaces on a single cluster using Astra Control. An app can contain resources spanning multiple namespaces within the same cluster. Astra Control does not support the ability for apps to be defined across multiple clusters.
- **Storage class**: If you install an application with a storage class explicitly set and you need to clone the app, the target cluster for the clone operation must 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.
- **Kubernetes resources**: Applications that use Kubernetes resources not collected by Astra Control might not have full app data management capabilities. Astra Control collects the following Kubernetes resources:

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

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) is fully supported. Managing apps installed with Helm 2 is not supported.
- **Operator-deployed apps**: Astra Control supports apps installed with namespace-scoped operators that are, in general, designed with a "pass-by-value" rather than "pass-by-reference" architecture. 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.

  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

  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.

Install apps on your cluster

After 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 one or more namespaces can be managed.

Define apps

After Astra Control discovers namespaces on your clusters, you can define applications that you want to manage. You can choose to manage an app spanning one or more namespaces or manage an entire namespace as a single application. It all comes down to the level of granularity that you need for data protection operations.

Although Astra Control enables you to separately manage both levels of the hierarchy (the namespace and the apps in that namespace or spanning namespaces), 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.

As an 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 as a single-namespace app.

Before you begin

- A Kubernetes cluster added to Astra Control.
- One or more installed apps on the cluster. Read more about supported app installation methods.
- Existing namespaces on the Kubernetes cluster that you added to Astra Control.
- (Optional) A Kubernetes label on any supported Kubernetes resources.

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.

About this task

- Before you begin, you should also understand managing standard and system namespaces.
- If you plan to use multiple namespaces with your apps in Astra Control, modify user roles with namespace constraints after you upgrade to an Astra Control Center version with multiple namespace support.
- For instructions on how to manage apps using the Astra Control API, see the Astra Automation and API information.

App management options

- Define resources to manage as an app
- Define a namespace to manage as an app
- (Tech preview) Define an application using a Kubernetes custom resource

Define resources to manage as an app

You can specify the Kubernetes resources that make up an app that you want to manage with Astra Control. Defining an app enables you to group elements of your Kubernetes cluster into a single app. This collection of Kubernetes resources is organized by namespace and label selector criteria.

Defining an app gives you more granular control over what to include in an Astra Control operation, including clone, snapshot, and backups.

When defining apps, ensure that you do not include a Kubernetes resource in multiple apps with protection policies. Overlapping protection policies on Kubernetes resources can cause data conflicts. Read more in an example.

Expand for more about adding cluster-scoped resources to your app namespaces.

You can import cluster resources that are associated with the namespace resources in addition to those Astra Control included automatically. You can add a rule that will include resources of a specific group, kind, version and optionally, label. You might want to do this if there are resources that Astra Control does not include automatically.

You cannot exclude any of the cluster-scoped resources that are automatically included by Astra Control.

You can add the following `apiVersions` (which are the groups combined with the API version):

Resource kind	apiVersions (group + version)
`ClusterRole`	rbac.authorization.k8s.io/v1
`ClusterRoleBinding`	rbac.authorization.k8s.io/v1
`CustomResource`	apiextensions.k8s.io/v1, apiextensions.k8s.io/v1beta1
`CustomResourceDefinition`	apiextensions.k8s.io/v1, apiextensions.k8s.io/v1beta1
`MutatingWebhookConfiguration`	admissionregistration.k8s.io/v1
`ValidatingWebhookConfiguration`	admissionregistration.k8s.io/v1

Steps

1. From the Applications page, select **Define**.
2. In the **Define application** window, enter the app name.
3. Choose the cluster on which your application is running in the **Cluster** drop-down list.
4. Choose a namespace for your application from the **Namespace** drop-down list.

   Apps can be defined within one or more specified namespaces on a single cluster using Astra Control. An app can contain resources spanning multiple namespaces within the same cluster. Astra Control does not support the ability for apps to be defined across multiple clusters.

5. (Optional) Enter a label for the Kubernetes resources in each namespace. You can specify a single label or label selector criteria (query).

   To learn more about Kubernetes labels, see the official Kubernetes documentation.

6. (Optional) Add additional namespaces for the app by selecting **Add namespace** and choosing the namespace from the drop-down list.
7. (Optional) Enter single label or label selector criteria for any additional namespaces you add.
8. (Optional) To include cluster-scoped resources in addition to those that Astra Control automatically includes, check **Include additional cluster-scoped resources** and complete the following:
   a. Select **Add include rule**.
   b. **Group**: From the drop-down list, select the API group of resources.

c. **Kind**: From the drop-down list, select the name of the object schema.

d. **Version**: Enter the API version.

e. **Label selector**: Optionally, include a label to add to the rule. This label is used to retrieve only those resources matching this label. If you don't provide a label, Astra Control collects all instances of the resource kind specified for that cluster.

f. Review the rule that is created based on your entries.

g. Select **Add**.

> You can create as many cluster-scoped resource rules as you want. The rules appear in the Define application Summary.

9. Select **Define**.
10. After you select **Define**, repeat the process for other apps, as needed.

After you finish defining an app, the app appears in `Healthy` state in the list of apps on the Applications page. You are now able to clone it and create backups and snapshots.

> 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.

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

To see the resources added to this app, select the **Resources** tab. Select the number after the resource name in the Resource column or enter the resource name in the Search to see the additional cluster-scoped resources included.

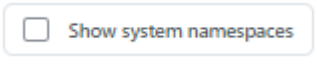
Define a namespace to manage as an app

You can add all Kubernetes resources in a namespace to Astra Control management by defining the resources of that namespace as an application. This method is preferable to defining apps individually if you intend to manage and protect all resources in a particular namespace in a similar way and at common intervals.

Steps

1. From the Clusters page, select a cluster.
2. Select the **Namespaces** tab.
3. Select the Actions menu for the namespace that contains the app resources you want to manage and select **Define as application**.

> If you want to define multiple applications, select from the namespaces list and select the **Actions** button in the upper-left corner and select **Define as application**. This will define multiple individual applications in their individual namespaces. For multi-namespace applications, see Define resources to manage as an app.

> Select the **Show system namespaces** checkbox to reveal system namespaces that are usually not used in app management by default. 
>
> Read more.

After the process completes, the applications that are associated with the namespace appear in the

`Associated applications` column.

[Tech preview] Define an application using a Kubernetes custom resource

You can specify the Kubernetes resources that you want to manage with Astra Control by defining them as an application using a custom resource (CR). You can add cluster-scoped resources if you want to manage those resources individually or all Kubernetes resources in a namespace if, for example, you intend to manage and protect all resources in a particular namespace in a similar way and at common intervals.

Steps

1. Create the custom resource (CR) file and name it (for example, `astra_mysql_app.yaml`).
2. Name the application in `metadata.name`.
3. Define application resources to be managed:

spec.includedClusterScopedResources

Include cluster-scoped resource types in addition to those that Astra Control automatically includes:

- **spec.includedClusterScopedResources**: *(Optional)* A list of cluster-scoped resource types to be included.
  - **groupVersionKind**: *(Optional)* Unambiguously identifies a kind.
    - **group**: *(Required if groupVersionKind is used)* API group of the resource to include.
    - **version**: *(Required if groupVersionKind is used)* API version of the resource to include.
    - **kind**: *(Required if groupVersionKind is used)* Kind of the resource to include.
  - **labelSelector**: *(Optional)* A label query for a set of resources. It's used to retrieve only those resources matching the label. If you don't provide a label, Astra Control collects all instances of the resource kind specified for that cluster. The result of matchLabels and matchExpressions are ANDed.
    - **matchLabels**: *(Optional)* A map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions that has a key field of "key", operator as "In", and values array containing only "value". The requirements are ANDed.
    - **matchExpressions**: *(Optional)* A list of label selector requirements. The requirements are ANDed.
      - **key**: *(Required if matchExpressions is used)* The label key associated with the label selector.
      - **operator**: *(Required if matchExpressions is used)* Represents a key's relationship to a set of values. Valid operators are `In`, `NotIn`, `Exists` and `DoesNotExist`.
      - **values**: *(Required if matchExpressions is used)_An array of string values. If the operator is `In` or `NotIn`, the values array must _not* be empty. If the operator is `Exists` or `DoesNotExist`, the values array must be empty.

spec.includedNamespaces

Include namespaces and resources within those resources in the application:

- **spec.includedNamespaces**: _(Required)_Defines the namespace and optional filters for resource selection.
  - **namespace**: *(Required)* The namespace that contains the app resources you want to manage with Astra Control.
  - **labelSelector**: *(Optional)* A label query for a set of resources. It's used to retrieve only those resources matching the label. If you don't provide a label, Astra Control collects all instances of the resource kind specified for that cluster. The result of matchLabels and matchExpressions are ANDed.
    - **matchLabels**: *(Optional)* A map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions that has a key field of "key", operator as "In", and values array containing only "value". The requirements are ANDed.
    - **matchExpressions**: *(Optional)* A list of label selector requirements. `key` and `operator` are required. The requirements are ANDed.
      - **key**: *(Required if matchExpressions is used)* The label key associated with the label selector.

- **operator**: *(Required if matchExpressions is used)* Represents a key's relationship to a set of values. Valid operators are `In`, `NotIn`, `Exists` and `DoesNotExist`.
- **values**: *(Required if matchExpressions is used)* An array of string values. If the operator is `In` or `NotIn`, the values array must *not* be empty. If the operator is `Exists` or `DoesNotExist`, the values array must be empty.

Example YAML:

```
apiVersion: astra.netapp.io/v1
kind: Application
metadata:
  name: astra_mysql_app
spec:
  includedNamespaces:
    - namespace: astra_mysql_app
      labelSelector:
        matchLabels:
          app: nginx
          env: production
        matchExpressions:
          - key: tier
            operator: In
            values:
              - frontend
              - backend
```

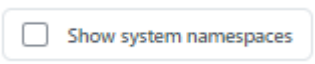
4. After you populate the `astra_mysql_app.yaml` file with the correct values, apply the CR:

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

What about system namespaces?

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

You can display system namespaces from the Namespaces tab for a selected cluster by selecting the **Show system namespaces** check box.



Astra Control Center is not shown by default as an application that you can manage, but you can back up and restore an Astra Control Center instance using another Astra Control Center instance.

Example: Separate Protection Policy for different releases

In this example, the devops team is managing a "canary" release deployment. The team's 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 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.

Find more information

- Use the Astra Control API
- Unmanage an app

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.

Additionally, you can replicate applications to a remote cluster in preparation for disaster recovery.

App protection workflow

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

[One] Protect 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] Adjust the protection policies

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

[Four] Replicate apps to a remote cluster

Replicate applications to a remote cluster by using NetApp SnapMirror technology. Astra Control replicates Snapshots to a remote cluster, providing asynchronous, disaster recovery capability.

[Five] In case of a disaster, restore your apps with the latest backup or replication to remote system

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). Or, you can use the replication to a remote system.

Protect apps with snapshots and backups

Protect all apps by taking snapshots and backups using an automated protection policy or on an ad-hoc basis. You can use the Astra Control Center UI or the Astra Control API to protect apps.

About this task

- **Helm deployed 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.
- **(OpenShift clusters only) Add policies**: 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
  ```

You can do the following tasks related to protecting your app data:

- Configure a protection policy
- Create a snapshot
- Create a backup
- Enable backup and restore for ontap-nas-economy operations
- Create an immutable backup
- View snapshots and backups
- Delete snapshots
- Cancel backups
- Delete backups

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. You can define a protection policy using either the Astra Control web UI or a custom resource (CR) file.

If you need backups or snapshots to run more frequently than once per hour, you can use the Astra Control REST API to create snapshots and backups.

If you are defining a protection policy that creates immutable backups to write once read many (WORM) buckets, ensure that the retention time for the backups is not shorter than the retention period configured for the bucket.

Offset backup and replication schedules to avoid schedule overlaps. For example, perform backups at the top of the hour every hour and schedule replication to start with a 5-minute offset and a 10-minute interval.

Configure a protection policy using the Web UI

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.

   When you set a retention level for backups, you can choose the bucket where you'd like to store the backups.

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

5. [**Tech preview**] Choose a destination bucket for the backups or snapshots from the list of storage buckets.
6. Select **Review**.
7. Select **Set Protection Policy.**

[Tech preview] Configure a protection policy using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-schedule-cr.yaml`. Update the values in brackets <> to match your Astra Control environment, cluster configuration, and data protection needs:
   - <CR_NAME>: The name of this custom resource; choose a unique and sensible name for your environment.
   - <APPLICATION_NAME>: The Kubernetes name of the application to back up.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents should be stored.
   - <BACKUPS_RETAINED>: The number of backups to retain. Zero indicates that no backups should be created.
   - <SNAPSHOTS_RETAINED>: The number of snapshots to retain. Zero indicates that no snapshots should be created.
   - <GRANULARITY>: The frequency at which the schedule should run. Possible values, along with required associated fields:
     - `hourly` (requires that you specify `spec.minute`)
     - `daily` (requires that you specify `spec.minute` and `spec.hour`)
     - `weekly` (requires that you specify `spec.minute`, `spec.hour`, and `spec.dayOfWeek`)
     - `monthly` (requires that you specify `spec.minute`, `spec.hour`, and `spec.dayOfMonth`)
   - <DAY_OF_MONTH>: *(Optional)* The day of the month (1 - 31) that the schedule should run. This field is required if the granularity is set to `monthly`.

- <DAY_OF_WEEK>: *(Optional)* The day of the week (0 - 7) that the schedule should run. Values of 0 or 7 indicate Sunday. This field is required if the granularity is set to `weekly`.
- <HOUR_OF_DAY>: *(Optional)* The hour of the day (0 - 23) that the schedule should run. This field is required if the granularity is set to `daily`, `weekly`, or `monthly`.
- <MINUTE_OF_HOUR>: *(Optional)* The minute of the hour (0 - 59) that the schedule should run. This field is required if the granularity is set to `hourly`, `daily`, `weekly`, or `monthly`.

```
apiVersion: astra.netapp.io/v1
kind: Schedule
metadata:
  namespace: astra-connector
  name: <CR_NAME>
spec:
  applicationRef: <APPLICATION_NAME>
  appVaultRef: <APPVAULT_NAME>
  backupRetention: "<BACKUPS_RETAINED>"
  snapshotRetention: "<SNAPSHOTS_RETAINED>"
  granularity: <GRANULARITY>
  dayOfMonth: "<DAY_OF_MONTH>"
  dayOfWeek: "<DAY_OF_WEEK>"
  hour: "<HOUR_OF_DAY>"
  minute: "<MINUTE_OF_HOUR>"
```

2. After you populate the `astra-control-schedule-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-schedule-cr.yaml
```

Result

Astra Control 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.

About this task

Astra Control supports snapshot creation using storage classes backed by the following drivers:

- `ontap-nas`
- `ontap-san`
- `ontap-san-economy`

If your app uses a storage class backed by the `ontap-nas-economy` driver, snapshots can't be created. Use an alternate storage class for snapshots.

Create a snapshot using the Web UI

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 **Next**.
4. [**Tech preview**] Choose a destination bucket for the snapshot from the list of storage buckets.
5. Review the snapshot summary and select **Snapshot**.

[Tech preview] Create a snapshot using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-snapshot-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this custom resource; choose a unique and sensible name for your environment.
   - <APPLICATION_NAME>: The Kubernetes name of the application to snapshot.
   - <APPVAULT_NAME>: The name of the AppVault where the snapshot contents should be stored.
   - <RECLAIM_POLICY>: *(Optional)* Defines what happens to a snapshot when the snapshot CR is deleted. Valid options:
     - `Retain`
     - `Delete` (default)

```
apiVersion: astra.netapp.io/v1
kind: Snapshot
metadata:
  namespace: astra-connector
  name: <CR_NAME>
spec:
  applicationRef: <APPLICATION_NAME>
  appVaultRef: <APPVAULT_NAME>
  reclaimPolicy: <RECLAIM_POLICY>
```

2. After you populate the `astra-control-snapshot-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-snapshot-cr.yaml
```

Result

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

Create a backup

You can back up an app at any time.

About this task

Buckets in Astra Control do not report available capacity. Before backing up or cloning apps managed by Astra Control, check bucket information in the appropriate storage management system.

If your app uses a storage class backed by the `ontap-nas-economy` driver, you need to enable backup and restore functionality. Be sure that you have defined a `backendType` parameter in your Kubernetes storage object with a value of `ontap-nas-economy` before performing any protection operations.

Astra Control supports backup creation using storage classes backed by the following drivers:

- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san`
- `ontap-san-economy`

Create a backup using the Web UI

Steps

1. Select **Applications**.
2. From the Options menu in the **Actions** column for the desired app, select **Back up**.
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. [**Tech preview**] Choose a destination bucket for the backup from the list of storage buckets.
6. Select **Next**.
7. Review the backup summary and select **Back up**.

[Tech preview] Create a backup using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-backup-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this custom resource; choose a unique and sensible name for your environment.
   - <APPLICATION_NAME>: The Kubernetes name of the application to back up.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents should be stored.

   ```
   apiVersion: astra.netapp.io/v1
   kind: Backup
   metadata:
     namespace: astra-connector
     name: <CR_NAME>
   spec:
     applicationRef: <APPLICATION_NAME>
     appVaultRef: <APPVAULT_NAME>
   ```

2. After you populate the `astra-control-backup-cr.yaml` file with the correct values, apply the CR:

   ```
   kubectl apply -f astra-control-backup-cr.yaml
   ```

Result

Astra Control 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.
- If you need to cancel a running backup, use the instructions in Cancel backups. To delete the backup, wait until it has completed and then use the instructions in Delete backups.
- 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.

Enable backup and restore for ontap-nas-economy operations

Astra Control Provisioner provides backup and restore functionality that can be enabled for storage backends that are using the `ontap-nas-economy` storage class.

Before you begin

- You have enabled Astra Control Provisioner.
- You have defined an application in Astra Control. This application will have limited protection functionality until you complete this procedure.
- You have `ontap-nas-economy` selected as the default storage class for your storage backend.

Steps

1. Do the following on the ONTAP storage backend:
   a. Find the SVM that is hosting the `ontap-nas-economy`-based volumes of the application.
   b. Log in to a terminal connected to ONTAP where the volumes are created.
   c. Hide the snapshot directory for the SVM:

      This change affects the entire SVM. The hidden directory will continue to be accessible.

      ```
      nfs modify -vserver <svm name> -v3-hide-snapshot enabled
      ```

      Verify that the snapshot directory on the ONTAP storage backend is hidden. Failure to hide this directory might lead to loss of access to your application, especially if it is using NFSv3.

2. Do the following in Astra Control Provisioner:
   a. Enable the snapshot directory for each PV that is `ontap-nas-economy` based and associated with the application:

      ```
      tridentctl update volume <pv name> --snapshot-dir=true --pool-level
      =true -n trident
      ```

   b. Confirm that the snapshot directory has been enabled for each associated PV:

```
tridentctl get volume <pv name> -n trident -o yaml | grep snapshotDir
```

Response:

```
snapshotDirectory: "true"
```

3. In Astra Control, refresh the application after enabling all associated snapshot directories so that Astra Control recognizes the changed value.

Result

The application is ready to backup and restore using Astra Control. Each PVC is also available to be used by other applications for backups and restores.

Create an immutable backup

An immutable backup cannot be modified, deleted, or overwritten as long as the retention policy on the bucket that stores the backup forbids it. You can create immutable backups by backing up applications to buckets that have a retention policy configured. Refer to Data protection for important information about working with immutable backups.

Before you begin

You need to configure the destination bucket with a retention policy. How you do this will differ depending on which storage provider you use. Refer to the storage provider documentation for more information:

- **Amazon Web Services**: Enable S3 Object Lock when creating the bucket and set a default retention mode of "governance" with a default retention period.
- **NetApp StorageGRID**: Enable S3 Object Lock when creating the bucket and set a default retention mode of "compliance" with a default retention period.

Buckets in Astra Control do not report available capacity. Before backing up or cloning apps managed by Astra Control, check bucket information in the appropriate storage management system.

If your app uses a storage class backed by the `ontap-nas-economy` driver, be sure that you have defined a `backendType` parameter in your Kubernetes storage object with a value of `ontap-nas-economy` before performing any protection operations.

Steps

1. Select **Applications**.
2. From the Options menu in the **Actions** column for the desired app, select **Back up**.
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 bucket for the backup from the list of storage buckets. A write once read many (WORM) bucket is indicated with a status of "Locked" next to the bucket name.

If the bucket is an unsupported type, this is indicated when you hover over or select the bucket.

6. Select **Next**.
7. Review the backup summary and select **Back up**.

Result

Astra Control creates an immutable 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.
- If you try to create two immutable backups of the same app to the same bucket at the same time, Astra Control prevents the second backup from starting. Wait until the first backup is complete before starting another.
- You cannot cancel a running immutable backup.
- 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.

An immutable backup is indicated with a status of "Locked" next to the bucket it is using.

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.

You cannot delete a snapshot that currently is being replicated.

Steps

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

Cancel backups

You can cancel a backup that is in progress.

To cancel a backup, the backup must be in `Running` state. You cannot cancel a backup that is in `Pending` state.

You cannot cancel a running immutable backup.

Steps

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 **Cancel**.
5. Type the word "cancel" to confirm the operation and then select **Yes, cancel backup**.

Delete backups

Delete the scheduled or on-demand backups that you no longer need. You cannot delete a backup made to an immutable bucket until the bucket's retention policy enables you to do so.

You cannot delete an immutable backup before the retention period expires.

If you need to cancel a running backup, use the instructions in Cancel backups. To delete the backup, wait until it has completed and then use these instructions.

Steps

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 deletes the backup.

[Tech preview] Protect an entire cluster

You can create a scheduled, automatic backup of any or all unmanaged namespaces on a cluster. These workflows are provided by NetApp as a Kubernetes service account, role bindings, and a cron job, orchestrated with a Python script.

How it works

When you configure and install the full-cluster backup workflow, a cron job runs periodically and protects any namespace that is not already managed, automatically creating protection policies based on schedules that you choose during installation.

If you don’t want to protect every unmanaged namespace on the cluster with the full cluster backup workflow, you can instead utilize the label-based backup workflow. The label-based backup workflow also uses a cron task, but instead of protecting all unmanaged namespaces, it identifies namespaces by labels you provide to optionally protect the namespaces based on bronze, silver, or gold backup policies.

When a new namespace is created that falls within the scope of your chosen workflow, it is automatically protected, without any administrator action. These workflows are implemented on a per-cluster basis, so different clusters can make use of either workflow with unique protection levels, depending on cluster importance.

Example: Full cluster protection

As an example, when you configure and install the full cluster backup workflow, any apps in any namespace are periodically managed and protected without further effort by the administrator. The namespace doesn’t have to exist at the time you install the workflow; if a namespace is added in the future, it will be protected.

Example: Label-based protection

For more granularity, you can use the label-based workflow. For example, you can install this workflow and tell your users to apply one of several labels to any namespaces they want to protect, depending on the level of protection they need. This enables users to create the namespace with one of those labels, and they don’t have to notify an administrator. Their new namespace and all apps within it are automatically protected.

Create a scheduled backup of all namespaces

You can create a scheduled backup of all namespaces on a cluster using the full cluster backup workflow.

Steps

1. Download the following files to a machine that has network access to your cluster:
   - components.yaml CRD file
   - protectCluster.py Python script
2. To configure and install the toolkit, follow the included instructions.

Create a scheduled backup of specific namespaces

You can create a scheduled backup of specific namespaces by their labels using the label-based backup workflow.

Steps

1. Download the following files to a machine that has network access to your cluster:
   - components.yaml CRD file
   - protectCluster.py Python script
2. To configure and install the toolkit, follow the included instructions.

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 Astra Control API to restore apps.

Before you begin

- **Protect your apps first**: It is strongly recommended that you take a snapshot or backup of your application before restoring it. This will enable you to clone from the snapshot or backup if the restore is unsuccessful.
- **Check destination volumes**: If you restore to a different storage class, ensure that the storage class uses the same persistent volume access mode (for example, ReadWriteMany). The restore operation will fail if the destination persistent volume access mode is different. For example, if your source persistent volume uses the RWX access mode, selecting a destination storage class that is not able to provide RWX, such as Azure Managed Disks, AWS EBS, Google Persistent Disk or `ontap-san`, will cause the restore operation to fail. For more information about persistent volume access modes, refer to the Kubernetes documentation.
- **Plan for space needs**: When you perform an in-place restore of an application that uses NetApp ONTAP storage, the space used by the restored app can double. After performing an in-place restore, remove any unwanted snapshots from the restored application to free up storage space.
- **(Red Hat OpenShift clusters only) Add policies**: 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
  ```

- **Supported storage class drivers**: Astra Control supports restoring backups using storage classes backed by the following drivers:
  - `ontap-nas`
  - `ontap-nas-economy`
  - `ontap-san`
  - `ontap-san-economy`
- **(ontap-nas-economy driver only) Backups and restores**: Before backing up or restoring an app that uses a storage class backed by the `ontap-nas-economy` driver, verify that the snapshot directory on the ONTAP storage backend is hidden. Failure to hide this directory might lead to loss of access to your application, especially if it is using NFSv3.
- **Helm deployed apps**: Apps deployed with Helm 3 (or upgraded from Helm 2 to Helm 3) are fully supported. Apps deployed with Helm 2 are not supported.

> Performing an in-place restore operation on an app that shares resources with another app can have unintended results. Any resources that are shared between the apps are replaced when an in-place restore is performed on one of the apps. For more information, see this example.

Perform the following steps, depending on the type of archive you want to restore:

Restore data from backup or snapshot using the web UI

You can restore data using the Astra Control web UI.

Steps

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

2. From the Options menu in the Actions column, select **Restore**.
3. Choose the restore type:
   - **Restore to original namespaces**: Use this procedure to restore the app in-place to the original cluster.

     > If your app uses a storage class backed by the `ontap-nas-economy` driver, you must restore the app using the original storage classes. You cannot specify a different storage class if you are restoring the app to the same namespace.

     a. Select the snapshot or backup to use to restore the app in-place, which reverts the app to an earlier version of itself.
     b. Select **Next**.

     > If you restore to a namespace that was previously deleted, a new namespace with the same name is created as part of the restore process. Any users that had rights to manage apps in the previously deleted namespace need to manually restore rights to the newly re-created namespace.

   - **Restore to new namespaces**: Use this procedure to restore the app to another cluster or with different namespaces from the source.
     a. Specify the name for the restored app.
     b. Choose the destination cluster for the app you intend to restore.
     c. Enter a destination namespace for each source namespace associated with the app.

     > Astra Control creates new destination namespaces as part of this restore option. Destination namespaces that you specify must not be already present on the destination cluster.

     d. Select **Next**.
     e. Select the snapshot or backup to use to restore the app.
     f. Select **Next**.
     g. Choose one of the following:
        - **Restore using original storage classes**: The application uses the originally associated storage class unless it does not exist on the target cluster. In this case, the default storage class for the cluster will be used.
        - **Restore using a different storage class**: Select a storage class that exists on the target cluster. All application volumes, regardless of their originally associated storage classes, will be migrated to this different storage class as part of the restore.
     h. Select **Next**.
4. Choose any resources to filter:
   - **Restore all resources**: Restore all resources associated with the original app.
   - **Filter resources**: Specify rules to restore a sub-set of the original application resources:
     i. Choose to include or exclude resources from the restored application.
     ii. Select either **Add include rule** or **Add exclude rule** and configure the rule to filter the correct resources during application restore. You can edit a rule or remove it and create a rule again until the configuration is correct.

To learn about configuring include and exclude rules, see Filter resources during an application restore.

5. Select **Next**.
6. Review details about the restore action carefully, type "restore" (if prompted), and select **Restore**.

[Tech preview] Restore from backup using a custom resource (CR)

You can restore data from a backup using a custom resource (CR) file either to a different namespace or the original source namespace.

Restore from backup using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-backup-restore-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this CR operation; choose a sensible name for your environment.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents are stored.
   - <BACKUP_PATH>: The path inside AppVault where the backup contents are stored. For example:

     ```
     ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
     20231213023800_94347756-9d9b-401d-a0c3
     ```

   - <SOURCE_NAMESPACE>: The source namespace of the restore operation.
   - <DESTINATION_NAMESPACE>: The destination namespace of the restore operation.

     ```
     apiVersion: astra.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: <CR_NAME>
       namespace: astra-connector
     spec:
       appVaultRef: <APPVAULT_NAME>
       appArchivePath: <BACKUP_PATH>
       namespaceMapping: [{"source": "<SOURCE_NAMESPACE>",
     "destination": "<DESTINATION_NAMESPACE>"}]
     ```

2. (Optional) If you need to select only certain resources of the application to restore, add filtering that includes or excludes resources marked with particular labels:
   - "<INCLUDE-EXCLUDE>": *(Required for filtering)* Use `include` or `exclude` to include or exclude a resource defined in resourceMatchers. Add the following resourceMatchers parameters to define the resources to be included or excluded:
     - <GROUP>: *(Optional)* Group of the resource to be filtered.
     - <KIND>: *(Optional)* Kind of the resource to be filtered.
     - <VERSION>: *(Optional)* Version of the resource to be filtered.
     - <NAMES>: *(Optional)* Names in the Kubernetes metadata.name field of the resource to be filtered.
     - <NAMESPACES>: *(Optional)* Namespaces in the Kubernetes metadata.name field of the resource to be filtered.
     - <SELECTORS>: *(Optional)* Label selector string in the Kubernetes metadata.name field of the resource as defined in Kubernetes documentation. Example: `"trident.netapp.io/os=linux"`.

       Example:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. After you populate the `astra-control-backup-restore-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-backup-restore-cr.yaml
```

Restore from backup to the original namespace using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-backup-ipr-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this CR operation; choose a sensible name for your environment.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents are stored.
   - <BACKUP_PATH>: The path inside AppVault where the backup contents are stored. For example:

```
ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
20231213023800_94347756-9d9b-401d-a0c3
```

```
apiVersion: astra.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: <CR_NAME>
  namespace: astra-connector
spec:
  appVaultRef: <APPVAULT_NAME>
  appArchivePath: <BACKUP_PATH>
```

2. (Optional) If you need to select only certain resources of the application to restore, add filtering that includes or excludes resources marked with particular labels:
   - "<INCLUDE-EXCLUDE>": *(Required for filtering)* Use `include` or `exclude` to include or exclude a resource defined in resourceMatchers. Add the following resourceMatchers parameters to define the resources to be included or excluded:

- <GROUP>: *(Optional)* Group of the resource to be filtered.
- <KIND>: *(Optional)* Kind of the resource to be filtered.
- <VERSION>: *(Optional)* Version of the resource to be filtered.
- <NAMES>: *(Optional)* Names in the Kubernetes metadata.name field of the resource to be filtered.
- <NAMESPACES>: *(Optional)* Namespaces in the Kubernetes metadata.name field of the resource to be filtered.
- <SELECTORS>: *(Optional)* Label selector string in the Kubernetes metadata.name field of the resource as defined in Kubernetes documentation. Example: `"trident.netapp.io/os=linux"`.

  Example:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. After you populate the `astra-control-backup-ipr-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-backup-ipr-cr.yaml
```

[Tech preview] Restore from snapshot using a custom resource (CR)

You can restore data from a snapshot using a custom resource (CR) file either to a different namespace or the original source namespace.

Restore from snapshot using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-snapshot-restore-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this CR operation; choose a sensible name for your environment.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents are stored.
   - <BACKUP_PATH>: The path inside AppVault where the backup contents are stored. For example:

     ```
     ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
     20231213023800_94347756-9d9b-401d-a0c3
     ```

   - <SOURCE_NAMESPACE>: The source namespace of the restore operation.
   - <DESTINATION_NAMESPACE>: The destination namespace of the restore operation.

     ```
     apiVersion: astra.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: <CR_NAME>
       namespace: astra-connector
     spec:
       appArchivePath: <BACKUP_PATH>
       appVaultRef: <APPVAULT_NAME>
       namespaceMapping: [{"source": "<SOURCE_NAMESPACE>",
     "destination": "<DESTINATION_NAMESPACE>"}]
     ```

2. (Optional) If you need to select only certain resources of the application to restore, add filtering that includes or excludes resources marked with particular labels:
   - "<INCLUDE-EXCLUDE>": *(Required for filtering)* Use `include` or `exclude` to include or exclude a resource defined in resourceMatchers. Add the following resourceMatchers parameters to define the resources to be included or excluded:
     - <GROUP>: *(Optional)* Group of the resource to be filtered.
     - <KIND>: *(Optional)* Kind of the resource to be filtered.
     - <VERSION>: *(Optional)* Version of the resource to be filtered.
     - <NAMES>: *(Optional)* Names in the Kubernetes metadata.name field of the resource to be filtered.
     - <NAMESPACES>: *(Optional)* Namespaces in the Kubernetes metadata.name field of the resource to be filtered.
     - <SELECTORS>: *(Optional)* Label selector string in the Kubernetes metadata.name field of the resource as defined in Kubernetes documentation. Example: `"trident.netapp.io/os=linux"`.

Example:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. After you populate the `astra-control-snapshot-restore-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-snapshot-restore-cr.yaml
```

Restore from snapshot to the original namespace using a CR

Steps

1. Create the custom resource (CR) file and name it `astra-control-snapshot-ipr-cr.yaml`. Update the values in brackets <> to match your Astra Control environment and cluster configuration:
   - <CR_NAME>: The name of this CR operation; choose a sensible name for your environment.
   - <APPVAULT_NAME>: The name of the AppVault where the backup contents are stored.
   - <BACKUP_PATH>: The path inside AppVault where the backup contents are stored. For example:

```
ONTAP-S3_1343ff5e-4c41-46b5-af00/backups/schedule-
20231213023800_94347756-9d9b-401d-a0c3
```

```
apiVersion: astra.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: <CR_NAME>
  namespace: astra-connector
spec:
  appArchivePath: <BACKUP_PATH>
  appVaultRef: <APPVAULT_NAME>
```

2. (Optional) If you need to select only certain resources of the application to restore, add filtering that includes or excludes resources marked with particular labels:
   - "<INCLUDE-EXCLUDE>": *(Required for filtering)* Use `include` or `exclude` to include or exclude

a resource defined in resourceMatchers. Add the following resourceMatchers parameters to define the resources to be included or excluded:

- <GROUP>: *(Optional)* Group of the resource to be filtered.
- <KIND>: *(Optional)* Kind of the resource to be filtered.
- <VERSION>: *(Optional)* Version of the resource to be filtered.
- <NAMES>: *(Optional)* Names in the Kubernetes metadata.name field of the resource to be filtered.
- <NAMESPACES>: *(Optional)* Namespaces in the Kubernetes metadata.name field of the resource to be filtered.
- <SELECTORS>: *(Optional)* Label selector string in the Kubernetes metadata.name field of the resource as defined in Kubernetes documentation. Example: `"trident.netapp.io/os=linux"`.

  Example:

```
spec:
    resourceFilter:
        resourceSelectionCriteria: "<INCLUDE-EXCLUDE>"
        resourceMatchers:
           group: <GROUP>
           kind: <KIND>
           version: <VERSION>
           names: <NAMES>
           namespaces: <NAMESPACES>
           labelSelectors: <SELECTORS>
```

3. After you populate the `astra-control-snapshot-ipr-cr.yaml` file with the correct values, apply the CR:

```
kubectl apply -f astra-control-snapshot-ipr-cr.yaml
```

Result

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

After a data protection operation (clone, backup, or restore) and subsequent persistent volume resize, there is a delay of up to twenty minutes before the new volume size is shown in the web 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.

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 clone or restore operation creates a new namespace, 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.

Filter resources during an application restore

You can add a filter rule to a restore operation that will specify existing application resources to be included or excluded from the restored application. You can include or exclude resources based on a specified namespace, label, or GVK (GroupVersionKind).

Expand for more about include and exclude scenarios

- **You select an include rule with original namespaces (in-place restore)**: Existing application resources that you define in the rule will be deleted and replaced by those from the selected snapshot or backup you are using for the restore. Any resources that you do not specify in the include rule will remain unchanged.
- **You select an include rule with new namespaces**: Use the rule to select the specific resources you want in the restored application. Any resources that you do not specify in the include rule will not be included in the restored application.
- **You select an exclude rule with original namespaces (in-place restore)**: The resources you specify to be excluded will not be restored and remain unchanged. Resources that you do not specify to exclude will be restored from the snapshot or backup. All data on persistent volumes will be deleted and recreated if the corresponding StatefulSet is part of the filtered resources.
- **You select an exclude rule with new namespaces**: Use the rule to select the specific resources you want to remove from the restored application. Resources that you do not specify to exclude will be restored from the snapshot or backup.

Rules are either include or exclude types. Rules combining resource inclusion and exclusion are not available.

Steps

1. After you have chosen to filter resources and selected an include or exclude option in the Restore App wizard, select **Add include rule** or **Add exclude rule**.

   You cannot exclude any cluster-scoped resources that are automatically included by Astra Control.

2. Configure the filter rule:

   You must specify at least one namespace, label, or GVK. Ensure that any resources you retain after the filter rules are applied are sufficient to keep the restored application in a healthy state.

   a. Select a specific namespace for the rule. If you don't make a selection, all namespaces will be used in the filter.

      If your application originally contained multiple namespaces and you restore it to new namespaces, all namespaces will be created even if they don't contain resources.

b. (Optional) Enter a resource name.

c. (Optional) **Label selector**: Include a label selector to add to the rule. The label selector is used to filter only those resources matching the selected label.

d. (Optional) Select **Use GVK (GroupVersionKind) set to filter resources** for additional filtering options.

> If you use a GVK filter, you must specify Version and Kind.

i. (Optional) **Group**: From the drop-down list, select the Kubernetes API group.

ii. **Kind**: From the drop-down list, select the object schema for the Kubernetes resource type to use in the filter.

iii. **Version**: Select the Kubernetes API version.

3. Review the rule that is created based on your entries.
4. Select **Add**.

> You can create as many resource include and exclude rules as you want. The rules appear in the restore application summary before you initiate the operation.

In-place restore complications for an app that shares resources with another app

You can perform an in-place restore operation on an app that shares resources with another app and produce unintended results. Any resources that are shared between the apps are replaced when an in-place restore is performed on one of the apps.

The following is an example scenario that creates an undesirable situation when using NetApp SnapMirror replication for a restore:

1. You define the application `app1` using the namespace `ns1`.
2. You configure a replication relationship for `app1`.
3. You define the application `app2` (on the same cluster) using the namespaces `ns1` and `ns2`.
4. You configure a replication relationship for `app2`.
5. You reverse replication for `app2`. This causes the `app1` app on the source cluster to be deactivated.

Replicate apps between storage backends using SnapMirror technology

Using Astra Control, you can build business continuity for your applications with a low-RPO (Recovery Point Objective) and low-RTO (Recovery Time Objective) using asynchronous replication capabilities of NetApp SnapMirror technology. Once configured, this enables your applications to replicate data and application changes from one storage backend to another, on the same cluster or between different clusters.

For a comparison between backups/restores and replication, refer to Data protection concepts.

You can replicate apps in different scenarios, such as the following on-premises only, hybrid, and multi-cloud scenarios:

- On-premises site A to on-premises site A

- On-premises site A to on-premises site B
- On-premises to cloud with Cloud Volumes ONTAP
- Cloud with Cloud Volumes ONTAP to on-premises
- Cloud with Cloud Volumes ONTAP to cloud (between different regions in the same cloud provider or to different cloud providers)

Astra Control can replicate apps across on-premises clusters, on-premises to cloud (using Cloud Volumes ONTAP) or between clouds (Cloud Volumes ONTAP to Cloud Volumes ONTAP).

> You can simultaneously replicate a different app in the opposite direction. For example, Apps A, B, C can be replicated from Datacenter 1 to Datacenter 2; and Apps X, Y, Z can be replicated from Datacenter 2 to Datacenter 1.

Using Astra Control, you can do the following tasks related to replicating applications:

- Set up a replication relationship
- Bring a replicated app online on the destination cluster (failover)
- Resync a failed over replication
- Reverse application replication
- Fail back applications to the original source cluster
- Delete an application replication relationship

Replication prerequisites

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

ONTAP clusters

- **Astra Control Provisioner or Astra Trident**: Astra Control Provisioner or Astra Trident must exist on both the source and destination Kubernetes clusters that utilize ONTAP as a backend. Astra Control supports replication with NetApp SnapMirror technology using storage classes backed by the following drivers:
  - `ontap-nas`
  - `ontap-san`
- **Licenses**: ONTAP SnapMirror asynchronous licenses using the Data Protection bundle must be enabled on both the source and destination ONTAP clusters. Refer to SnapMirror licensing overview in ONTAP for more information.

Peering

- **Cluster and SVM**: The ONTAP storage backends must be peered. Refer to Cluster and SVM peering overview for more information.

  > Ensure that the SVM names used in the replication relationship between two ONTAP clusters are unique.

- **Astra Control Provisioner or Astra Trident and SVM**: The peered remote SVMs must be available to Astra Control Provisioner or Astra Trident on the destination cluster.

Astra Control Center

Deploy Astra Control Center in a third fault domain or secondary site for seamless disaster recovery.

- **Managed backends**: You need to add and manage ONTAP storage backends in Astra Control Center to create a replication relationship.

  Adding and managing ONTAP storage backends in Astra Control Center is optional if you have enabled Astra Control Provisioner.

- **Managed clusters**: Add and manage the following clusters with Astra Control, ideally at different failure domains or sites:
  - Source Kubernetes cluster
  - Destination Kubernetes cluster
  - Associated ONTAP clusters
- **User accounts**: When you add an ONTAP storage backend to Astra Control Center, apply user credentials with the "admin" role. This role has access methods `http` and `ontapi` enabled on both ONTAP source and destination clusters. Refer to Manage User Accounts in ONTAP documentation for more information.

  With Astra Control Provisioner functionality, you don't need to specifically define an "admin" role to manage clusters in Astra Control Center as these credentials aren't required by Astra Control Center.

Astra Control Center does not support NetApp SnapMirror replication for storage backends that are using the NVMe over TCP protocol.

Astra Trident / ONTAP configuration

Astra Control Center requires that you configure at least one storage backend that supports replication for both the source and destination clusters. If the source and destination clusters are the same, the destination application should use a different storage backend than the source application for the best resiliency.

Astra Control replication supports apps that use a single storage class. When you add an app to a namespace, be sure the app has the same storage class as other apps in the namespace. When you add a PVC to a replicated app, be sure the new PVC has the same storage class as other PVCs in the namespace.

Set up a replication relationship

Setting up a replication relationship involves the following:

- Choosing how frequently you want Astra Control to take an app snapshot (which includes the app's Kubernetes resources as well as the volume snapshots for each of the app's volumes)
- Choosing the replication schedule (included Kubernetes resources as well as persistent volume data)
- Setting the time for the snapshot to be taken

Steps

1. From the Astra Control left navigation, select **Applications**.

2. Select the **Data Protection** > **Replication** tab.
3. Select **Configure replication policy**. Or, from the Application Protection box, select the Actions option and select **Configure replication policy**.
4. Enter or select the following information:
    - **Destination cluster**: Enter a destination cluster (this can be the same as the source cluster).
    - **Destination storage class**: Select or enter the storage class that uses the peered SVM on the destination ONTAP cluster. As a best practice, the destination storage class should point to a different storage backend than the source storage class.
    - **Replication type**: `Asynchronous` is currently the only replication type available.
    - **Destination namespace**: Enter new or existing destination namespaces for the destination cluster.
    - (Optional) Add additional namespaces by selecting **Add namespace** and choosing the namespace from the drop-down list.
    - **Replication frequency**: Set how often you want Astra Control to take a snapshot and replicate it to the destination.
    - **Offset**: Set the number of minutes from the top of the hour that you want Astra Control to take a snapshot. You might want to use an offset so that it doesn't coincide with other scheduled operations.

    Offset backup and replication schedules to avoid schedule overlaps. For example, perform backups at the top of the hour every hour and schedule replication to start with a 5-minute offset and a 10-minute interval.

5. Select **Next**, review the summary, and select **Save**.

    At first, the status displays "app-mirror" before the first schedule occurs.

    Astra Control creates an application snapshot used for replication.

6. To see the application snapshot status, select the **Applications** > **Snapshots** tab.

    The snapshot name uses the format of `replication-schedule-<string>`. Astra Control retains the last snapshot that was used for replication. Any older replication snapshots are deleted after successful completion of replication.

Result

This creates the replication relationship.

Astra Control completes the following actions as a result of establishing the relationship:

- Creates a namespace on the destination (if it doesn't exist)
- Creates a PVC on the destination namespace corresponding to the source app's PVCs.
- Takes an initial app-consistent snapshot.
- Establishes the SnapMirror relationship for persistent volumes using the initial snapshot.

The **Data Protection** page shows the replication relationship state and status:
<Health status> | <Relationship life cycle state>

For example:

Normal | Established

Learn more about replication states and status at the end of this topic.

Bring a replicated app online on the destination cluster (failover)

Using Astra Control, you can fail over replicated applications to a destination cluster. This procedure stops the replication relationship and brings the app online on the destination cluster. This procedure does not stop the app on the source cluster if it was operational.

Steps

1. From the Astra Control left navigation, select **Applications**.
2. Select the **Data Protection** > **Replication** tab.
3. From the Actions menu, select **Fail over**.
4. In the Fail over page, review the information and select **Fail over**.

Result

The following actions occur as a result of the failover procedure:

- The destination app is started based on the latest replicated snapshot.
- The source cluster and app (if operational) are not stopped and will continue to run.
- The replication state changes to "Failing over" and then to "Failed over" when it has completed.
- The source app's protection policy is copied to the destination app based on the schedules present on the source app at the time of the failover.
- If the source app has one or more post-restore execution hooks enabled, those execution hooks are run for the destination app.
- Astra Control shows the app both on the source and destination clusters and its respective health.

Resync a failed over replication

The resync operation re-establishes the replication relationship. You can choose the source of the relationship to retain the data on the source or destination cluster. This operation re-establishes the SnapMirror relationships to start the volume replication in the direction of choice.

The process stops the app on the new destination cluster before re-establishing replication.

During the resync process, the life cycle state shows as "Establishing."

Steps

1. From the Astra Control left navigation, select **Applications**.
2. Select the **Data Protection** > **Replication** tab.
3. From the Actions menu, select **Resync**.
4. In the Resync page, select either the source or destination app instance containing the data that you want to preserve.

Choose the resync source carefully, as the data on the destination will be overwritten.

5. Select **Resync** to continue.

6. Type "resync" to confirm.
7. Select **Yes, resync** to finish.

Result

- The Replication page shows "Establishing" as the replication status.
- Astra Control stops the application on the new destination cluster.
- Astra Control re-establishes the persistent volume replication in the selected direction using SnapMirror resync.
- The Replication page shows the updated relationship.

Reverse application replication

This is the planned operation to move the application to the destination storage backend while continuing to replicate back to the original source storage backend. Astra Control stops the source application and replicates the data to the destination before failing over to the destination app.

In this situation, you are swapping the source and destination.

Steps

1. From the Astra Control left navigation, select **Applications**.
2. Select the **Data Protection** > **Replication** tab.
3. From the Actions menu, select **Reverse replication**.
4. In the Reverse Replication page, review the information and select **Reverse replication** to continue.

Result

The following actions occur as a result of the reverse replication:

- A snapshot is taken of the original source app's Kubernetes resources.
- The original source app's pods are gracefully stopped by deleting the app's Kubernetes resources (leaving PVCs and PVs in place).
- After the pods are shut down, snapshots of the app's volumes are taken and replicated.
- The SnapMirror relationships are broken, making the destination volumes ready for read/write.
- The app's Kubernetes resources are restored from the pre-shutdown snapshot, using the volume data replicated after the original source app was shut down.
- Replication is re-established in the reverse direction.

Fail back applications to the original source cluster

Using Astra Control, you can achieve "fail back" after a failover operation by using the following sequence of operations. In this workflow to restore the original replication direction, Astra Control replicates (resyncs) any application changes back to the original source application before reversing the replication direction.

This process starts from a relationship that has completed a failover to a destination and involves the following steps:

- Start with a failed over state.
- Resync the relationship.

- Reverse the replication.

Steps

1. From the Astra Control left navigation, select **Applications**.
2. Select the **Data Protection** > **Replication** tab.
3. From the Actions menu, select **Resync**.
4. For a fail back operation, choose the failed over app as the source of the resync operation (preserving any data written post failover).
5. Type "resync" to confirm.
6. Select **Yes, resync** to finish.
7. After the resync is complete, in the Data Protection > Replication tab, from the Actions menu, select **Reverse replication**.
8. In the Reverse Replication page, review the information and select **Reverse replication**.

Result

This combines the results from the "resync" and "reverse relationship" operations to bring the application online on the original source cluster with replication resumed to the original destination cluster.

Delete an application replication relationship

Deleting the relationship results in two separate apps with no relationship between them.

Steps

1. From the Astra Control left navigation, select **Applications**.
2. Select the **Data Protection** > **Replication** tab.
3. From the Application Protection box or in the relationship diagram, select **Delete replication relationship**.

Result

The following actions occur as a result of deleting a replication relationship:

- If the relationship is established but the app has not yet been brought online on the destination cluster (failed over), Astra Control retains PVCs created during initialization, leaves an "empty" managed app on the destination cluster, and retains the destination app to keep any backups that might have been created.
- If the app has been brought online on the destination cluster (failed over), Astra Control retains PVCs and destination apps. Source and destination apps are now treated as independent apps. The backup schedules remain on both apps but are not associated with each other.

Replication relationship health status and relationship life cycle states

Astra Control displays the health of the relationship and the states of the life cycle of the replication relationship.

Replication relationship health statuses

The following statuses indicate the health of the replication relationship:

- **Normal**: The relationship is either establishing or has established, and the most recent snapshot transferred successfully.

- **Warning**: The relationship is either failing over or has failed over (and therefore is no longer protecting the source app).
- **Critical**
  - The relationship is establishing or failed over, and the last reconcile attempt failed.
  - The relationship is established, and the last attempt to reconcile the addition of a new PVC is failing.
  - The relationship is established (so a successful snapshot has replicated, and failover is possible), but the most recent snapshot failed or failed to replicate.

Replication life cycle states

The following states states reflect the different stages of the replication life cycle:

- **Establishing**: A new replication relationship is being created. Astra Control creates a namespace if needed, creates persistent volume claims (PVCs) on new volumes on the destination cluster, and creates SnapMirror relationships. This status can also indicate that the replication is resyncing or reversing replication.
- **Established**: A replication relationship exists. Astra Control periodically checks that the PVCs are available, checks the replication relationship, periodically creates snapshots of the app, and identifies any new source PVCs in the app. If so, Astra Control creates the resources to include them in the replication.
- **Failing over**: Astra Control breaks the SnapMirror relationships and restores the app's Kubernetes resources from the last successfully replicated app snapshot.
- **Failed over**: Astra Control stops replicating from the source cluster, uses the most recent (successful) replicated app snapshot on the destination, and restores the Kubernetes resources.
- **Resyncing**: Astra Control resyncs the new data on the resync source to the resync destination by using SnapMirror resync. This operation might overwrite some of the data on the destination based on the direction of the sync. Astra Control stops the app running on the destination namespace and removes the Kubernetes app. During the resyncing process, the status shows as "Establishing."
- **Reversing**: The is the planned operation to move the application to the destination cluster while continuing to replicate back to the original source cluster. Astra Control stops the application on the source cluster, replicates the data to the destination before failing over the app to the destination cluster. During the reverse replication, the status shows as "Establishing."
- **Deleting**:
  - If the replication relationship was established but not failed over yet, Astra Control removes PVCs that were created during replication and deletes the destination managed app.
  - If the replication failed over already, Astra Control retains the PVCs and destination app.

Clone and migrate apps

Astra Control Center UI or Astra Control API to clone and migrate apps.

You can clone an existing app to create a duplicate app on the same Kubernetes cluster or on another cluster. When Astra Control 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 Control Center UI or Astra Control API to clone and migrate apps.

Before you begin

- **Check destination volumes**: If you clone to a different storage class, ensure that the storage class uses

the same persistent volume access mode (for example, ReadWriteMany). The clone operation will fail if the destination persistent volume access mode is different. For example, if your source persistent volume uses the RWX access mode, selecting a destination storage class that is not able to provide RWX, such as Azure Managed Disks, AWS EBS, Google Persistent Disk or `ontap-san`, will cause the clone operation to fail. For more information about persistent volume access modes, refer to the Kubernetes documentation.

- To clone apps to a different cluster, you need to make sure the cloud instances containing the source and destination clusters (if they are not the same) have a default bucket. You'll need to assign a default bucket for each cloud instance.
- During clone operations, apps that need an IngressClass resource or webhooks to function properly must not have those resources already defined on the destination cluster.

> ⓘ 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`

Clone limitations

- **Explicit storage classes**: If you deploy an app with a storage class explicitly set and you need to clone the app, the target cluster must 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.
- **ontap-nas-economy-backed applications**: You can't use clone operations if your application's storage class is backed by the `ontap-nas-economy` driver. You can, however, enable backup and restore for ontap-nas-economy operations.
- **Clones and user constraints**: 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 clone or restore operation creates a new namespace, 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.
- **Clones use default buckets**: 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.
- **With Jenkins CI**: 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.
- **With S3 buckets**: 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.
- **With 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.

OpenShift considerations

- **Clusters and OpenShift versions**: 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.

- **Projects and UIDs**: 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. Specify details for the clone:
   - Enter a name.
   - Choose a destination cluster for the clone.
   - Enter destination namespaces for the clone. Each source namespace associated with the app maps to the destination namespace you define.

     Astra Control creates new destination namespaces as part of the clone operation. Destination namespaces that you specify must not be already present on the destination cluster.

   - Select **Next**.
   - Choose to keep the original storage class associated with the app or select a different storage class.

     You can migrate an app's storage class to a native cloud provider storage class or other supported storage class, migrate an app from a storage class backed by `ontap-nas-economy` to a storage class backed by `ontap-nas` on the same cluster, or copy the app to another cluster with a storage class backed by the `ontap-nas-economy` driver.

     If you select a different storage class and this storage class doesn't exist at the moment of restore, an error will be returned.

5. Select **Next**.
6. Review the information about the clone and select **Clone**.

Result

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

After a clone or restore operation creates a new namespace, 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.

After a data protection operation (clone, backup, or 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 action that you can configure to run in conjunction with a data protection operation of a managed app. For example, if you have a database app, you can use an execution hook to pause all database transactions before a snapshot, and resume transactions after the snapshot is complete. This ensures application-consistent snapshots.

Types of execution hooks

Astra Control Center supports the following types of execution hooks, based on when they can be run:

- Pre-snapshot
- Post-snapshot
- Pre-backup
- Post-backup
- Post-restore
- Post-failover

Execution hook filters

When you add or edit an execution hook to an application, you can add filters to an execution hook to manage which containers the hook will match. Filters are useful for applications that use the same container image on all containers, but might use each image for a different purpose (such as Elasticsearch). Filters allow you to create scenarios where execution hooks run on some but not necessarily all identical containers. If you create multiple filters for a single execution hook, they are combined with a logical AND operator. You can have up to 10 active filters per execution hook.

Each filter you add to an execution hook uses a regular expression to match containers in your cluster. When a hook matches a container, the hook will run its associated script on that container. Regular expressions for filters use the Regular Expression 2 (RE2) syntax, which does not support creating a filter that excludes containers from the list of matches. For information on the syntax that Astra Control supports for regular expressions in execution hook filters, see Regular Expression 2 (RE2) syntax support.

If you add a namespace filter to an execution hook that runs after a restore or clone operation and the restore or clone source and destination are in different namespaces, the namespace filter is only applied to the destination namespace.

Important notes about custom execution hooks

Consider the following when planning execution hooks for your apps.

Because 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.
If you start a backup or snapshot operation with associated execution hooks but then cancel it, the hooks are still allowed to run if the backup or snapshot operation has already begun. This means that the logic used in a post-backup execution hook cannot assume that the backup was completed.

- The execution hooks feature is disabled by default for new Astra Control deployments.
  - You need to enable the execution hooks feature before you can use execution hooks.
  - Owner or Admin users can enable or disable the execution hooks feature for all users defined in the current Astra Control account. Refer to Enable the execution hooks feature and Disable the execution hooks feature for instructions.
  - The feature enablement status is preserved during Astra Control upgrades.
- An execution hook must use a script to perform actions. Many execution hooks can reference the same script.
- Astra Control requires the scripts that execution hooks use to be written in the format of executable shell scripts.
- Script size is limited to 96KB.
- Astra Control uses execution hook settings and any matching criteria to determine which hooks are applicable to a snapshot, backup, or restore operation.
- All execution hook failures are soft failures; other hooks and the data protection operation 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.
- For on-demand data protection operations, all hook events are generated and saved in the **Activity** page event log. However, for scheduled data protection operations, only hook failure events are recorded in the event log (events generated by the scheduled data protection operations themselves are still recorded).
- If Astra Control Center fails over a replicated source app to the destination app, any post-failover execution hooks that are enabled for the source app are run for the destination app after the failover is complete.

If you have been running post-restore hooks with Astra Control Center 23.04 and upgraded your Astra Control Center to 23.07 or later, post-restore execution hooks will no longer be executed after a failover replication. You need to create new post-failover execution hooks for your apps. Alternatively, you can change the operation type of existing post-restore hooks intended for failovers from "post-restore" to "post-failover".

Order of execution

When a data protection operation is run, execution hook events take place in the following order:

1. Any applicable custom pre-operation execution hooks are run on the appropriate containers. You can create and run as many custom pre-operation hooks as you need, but the order of execution of these hooks before the operation is neither guaranteed nor configurable.

2. The data protection operation is performed.
3. Any applicable custom post-operation execution hooks are run on the appropriate containers. You can create and run as many custom post-operation hooks as you need, but the order of execution of these hooks after the operation is neither guaranteed nor configurable.

If you create multiple execution hooks of the same type (for example, pre-snapshot), the order of execution of those hooks is not guaranteed. However, the order of execution of hooks of different types is guaranteed. For example, the order of execution of a configuration that has all of the different types of hooks would look like this:

1. Pre-backup hooks executed
2. Pre-snapshot hooks executed
3. Post-snapshot hooks executed
4. Post-backup hooks executed
5. Post-restore hooks executed

You can see an example of this configuration in scenario number 2 from the table in Determine whether a hook will run.

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 and backups to ensure they are consistent. You can do this by cloning the app to a temporary namespace, restoring the snapshot or backup, and then testing the app.

Determine whether a hook will run

Use the following table to help determine if a custom execution hook will run for your app.

Note that all high-level app operations consist of running one of the basic operations of snapshot, backup, or restore. Depending on the scenario, a clone operation can consists of various combinations of these operations, so what execution hooks a clone operation runs will vary.

In-place restore operations require an existing snapshot or backup, so these operations don’t run snapshot or backup hooks.

If you start but then cancel a backup that includes a snapshot and there are associated execution hooks, some hooks might run, and others might not. This means that a post-backup execution hook cannot assume that the backup was completed. Keep in mind the following points for cancelled backups with associated execution hooks:

- The pre-backup and post-backup hooks are always run.
- If the backup includes a new snapshot and the snapshot has started, the pre-snapshot and post-snapshot hooks are run.
- If the backup is cancelled prior to the snapshot starting, the pre-snapshot and post-snapshot hooks are not run.

Scenario	Operation	Existing snapshot	Existing backup	Namespace	Cluster	Snapshot hooks run	Backup hooks run	Restore hooks run	Failover hooks run
1	Clone	N	N	New	Same	Y	N	Y	N
2	Clone	N	N	New	Different	Y	Y	Y	N
3	Clone or restore	Y	N	New	Same	N	N	Y	N
4	Clone or restore	N	Y	New	Same	N	N	Y	N
5	Clone or restore	Y	N	New	Different	N	N	Y	N
6	Clone or restore	N	Y	New	Different	N	N	Y	N
7	Restore	Y	N	Existing	Same	N	N	Y	N
8	Restore	N	Y	Existing	Same	N	N	Y	N
9	Snapshot	N/A	N/A	N/A	N/A	Y	N/A	N/A	N
10	Backup	N	N/A	N/A	N/A	Y	Y	N/A	N
11	Backup	Y	N/A	N/A	N/A	N	N	N/A	N
12	Failover	Y	N/A	Created by replication	Different	N	N	N	Y
13	Failover	Y	N/A	Created by replication	Same	N	N	N	Y

Execution hook examples

Visit the NetApp Verda GitHub project to download real execution hooks for popular apps such as Apache Cassandra and Elasticsearch. You can also see examples and get ideas for structuring your own custom execution hooks.

Enable the execution hooks feature

If you are an Owner or Admin user, you can enable the execution hooks feature. When you enable the feature, all users defined in this Astra Control account can use execution hooks and view existing execution hooks and hook scripts.

Steps

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

   The **Account** > **Feature settings** tab appears.

4. In the **Execution hooks** pane, select the settings menu.
5. Select **Enable**.
6. Note the security warning that appears.
7. Select **Yes, enable execution hooks**.

Disable the execution hooks feature

If you are an Owner or Admin user, you can disable the execution hooks feature for all users defined in this Astra Control account. You must delete all existing execution hooks before you can disable the execution hooks feature. Refer to Delete an execution hook for instructions on deleting an existing execution hook.

Steps

1. Go to **Account** and then select the **Feature settings** tab.
2. Select the **Execution hooks** tab.
3. In the **Execution hooks** pane, select the settings menu.
4. Select **Disable**.
5. Note the warning that appears.
6. Type `disable` to confirm that you want to disable the feature for all users.
7. Select **Yes, disable**.

View existing execution hooks

You can view existing custom 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, how many containers it matches, creation time, and when it runs (pre- or post-operation). You can select the + icon next to the hook name to expand the list of containers it will run on. To view event logs surrounding execution hooks for this application, go to the **Activity** tab.

View existing scripts

You can view the existing uploaded scripts. You can also see which scripts are in use, and what hooks are using them, on this page.

Steps

1. Go to **Account**.
2. Select the **Scripts** tab.

   You can see a list of existing uploaded scripts on this page. The **Used by** column shows which execution hooks are using each script.

Add a script

Each execution hook must use a script to perform actions. You can add one or more scripts that execution hooks can reference. Many execution hooks can reference the same script; this allows you to update many execution hooks by only changing one script.

Steps

1. Ensure that the execution hooks feature is enabled.
2. Go to **Account**.
3. Select the **Scripts** tab.
4. Select **Add**.
5. 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.
     e. Select **Save script**.
   - Paste in a custom script from the clipboard.
     a. Select the **Paste or type** 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.
6. Select **Save script**.

Result

The new script appears in the list on the **Scripts** tab.

Delete a script

You can remove a script from the system if it is no longer needed and not used by any execution hooks.

Steps

1. Go to **Account**.
2. Select the **Scripts** tab.
3. Choose a script you want to remove, and select the menu in the **Actions** column.
4. Select **Delete**.

> If the script is associated with one or more execution hooks, the **Delete** action is unavailable. To delete the script, first edit the associated execution hooks and associate them with a different script.

Create a custom execution hook

You can create a custom execution hook for an app and add it to Astra Control. Refer to 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 specific commands or providing the full path to an executable.

Steps

1. Ensure that the execution hooks feature is enabled.
2. Select **Applications** and then select the name of a managed app.
3. Select the **Execution hooks** tab.
4. Select **Add**.
5. In the **Hook Details** area:
   a. Determine when the hook should run by selecting an operation type from the **Operation** drop-down menu.
   b. Enter a unique name for the hook.
   c. (Optional) Enter any arguments to pass to the hook during execution, pressing the Enter key after each argument you enter to record each one.
6. (Optional) In the **Hook Filter Details** area, you can add filters to control which containers the execution hook runs on:
   a. Select **Add filter**.
   b. In the **Hook filter type** column, choose an attribute on which to filter from the drop-down menu.
   c. In the **Regex** column, enter a regular expression to use as the filter. Astra Control uses the Regular Expression 2 (RE2) regex syntax.

      > If you filter on the exact name of an attribute (such as a pod name) with no other text in the regular expression field, a substring match is performed. To match an exact name and only that name, use the exact string match syntax (for example, `^exact_podname$`).

   d. To add more filters, select **Add filter**.

      > Multiple filters for an execution hook are combined with a logical AND operator. You can have up to 10 active filters per execution hook.

7. When done, select **Next**.
8. In the **Script** area, do one of the following:
   - Add a new script.
     a. Select **Add**.
     b. Do one of the following:
        - Upload a custom script.
          i. Select the **Upload file** option.

ii. Browse to a file and upload it.
iii. Give the script a unique name.
iv. (Optional) Enter any notes other administrators should know about the script.
v. Select **Save script**.

- Paste in a custom script from the clipboard.
  i. Select the **Paste or type** option.
  ii. Select the text field and paste the script text into the field.
  iii. Give the script a unique name.
  iv. (Optional) Enter any notes other administrators should know about the script.

◦ Select an existing script from the list.

This instructs the execution hook to use this script.

9. Select **Next**.
10. Review the execution hook configuration.
11. Select **Add**.

Check the state of an execution hook

After a snapshot, backup, or restore operation finishes running, you can check the state of execution hooks that ran as part of the operation. You can use this status information to determine if you want to keep the execution hook, modify it, or delete it.

Steps

1. Select **Applications** and then select the name of a managed app.
2. Select the **Data protection** tab.
3. Select **Snapshots** to see running snapshots, or **Backups** to see running backups.

   The **Hook state** shows the status of the execution hook run after the operation is complete. You can hover over the state for more details. For example, if there are execution hook failures during a snapshot, hovering over the hook state for that snapshot gives a list of failed execution hooks. To see reasons for each failure, you can check the **Activity** page in the left-side navigation area.

View script usage

You can see which execution hooks use a particular script in the Astra Control web UI.

Steps

1. Select **Account**.
2. Select the **Scripts** tab.

   The **Used by** column in the list of scripts contains details on which hooks are using each script in the list.

3. Select the information in the **Used by** column for a script you are interested in.

   A more detailed list appears, with the names of hooks that are using the script and the type of operation they are configured to run with.

Edit an execution hook

You can edit an execution hook if you want to change its attributes, filters, or the script that it uses. You need to have Owner, Admin, or Member permissions to edit 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 edit.
4. Select **Edit**.
5. Make any needed changes, selecting **Next** after you complete each section.
6. Select **Save**.

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**.
5. In the resulting dialog, type "delete" to confirm.
6. Select **Yes, delete execution hook**.

For more information

- NetApp Verda GitHub project

Protect Astra Control Center using Astra Control Center

To better ensure resiliency against fatal errors on the Kubernetes cluster where Astra Control Center is running, protect the Astra Control Center application itself. You can backup and restore Astra Control Center using a secondary Astra Control Center instance or use Astra replication if the underlying storage is using ONTAP.

In these scenarios, a second instance of Astra Control Center is deployed and configured in a different fault domain and runs on a different second Kubernetes cluster than the primary Astra Control Center instance. The second Astra Control instance is used to back up and potentially restore the primary Astra Control Center instance. A restored or replicated Astra Control Center instance will continue to provide application data management for the application cluster applications and restore accessibility to backups and snapshots of those applications.

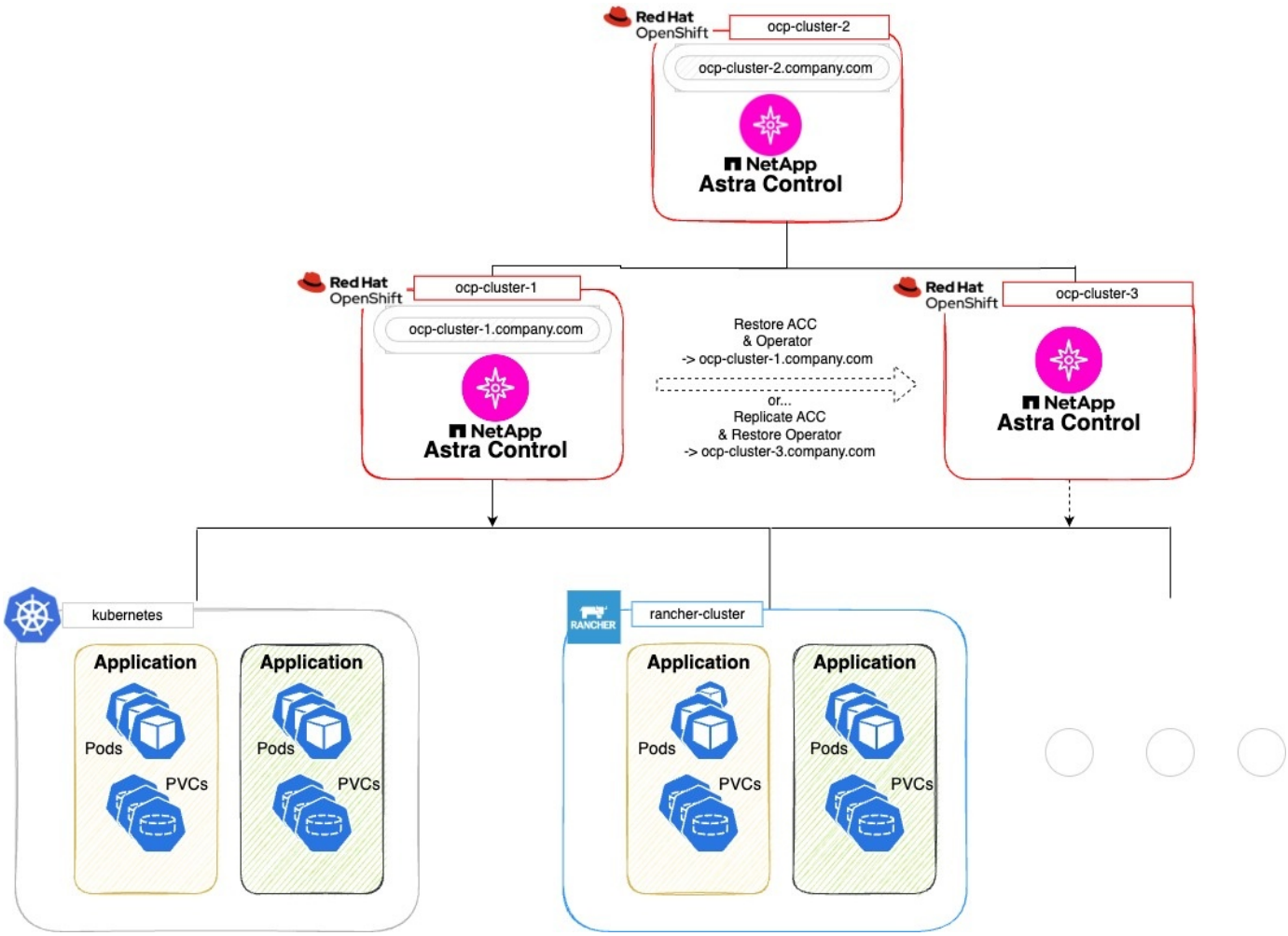
Before you begin

Ensure that you have the following before setting up protection scenarios for Astra Control Center:

- **A Kubernetes cluster running the primary Astra Control Center instance**: This cluster hosts the primary Astra Control Center instance which manages application clusters.
- **A second Kubernetes cluster of the same Kubernetes distribution type as the primary that is running the secondary Astra Control Center instance**: This cluster hosts the Astra Control Center instance that manages the primary Astra Control Center instance.
- **A third Kubernetes cluster of the same Kubernetes distribution type as the primary**: This cluster will host the restored or replicated instance of Astra Control Center. It must have the same Astra Control Center namespace available that is currently deployed on the primary. For example, if Astra Control Center is deployed in namespace `netapp-acc` on the source cluster, the namespace `netapp-acc` must be available and not used by any applications on the destination Kubernetes cluster.
- **S3-compatible buckets**: Each Astra Control Center instance has an accessible S3-compatible object storage bucket.
- **A configured load balancer**: The load balancer provides an IP address for Astra and must have network connectivity to the application clusters and both S3 buckets.
- **Clusters meet Astra Control Center requirements**: Each cluster used in Astra Control Center protection meets general Astra Control Center requirements.

About this task

These procedures describe the necessary steps to restore Astra Control Center to a new cluster using either backup and restore or replication. Steps are based on the example configuration depicted here:



In this example configuration, the following is shown:

- **A Kubernetes cluster running the primary Astra Control Center instance**:
  - OpenShift cluster: `ocp-cluster-1`
  - Astra Control Center primary instance: `ocp-cluster-1.company.com`
  - This cluster manages the application clusters.
- **The second Kubernetes cluster of the same Kubernetes distribution type as the primary that is running the secondary Astra Control Center instance**:
  - OpenShift cluster: `ocp-cluster-2`
  - Astra Control Center secondary instance: `ocp-cluster-2.company.com`
  - This cluster will be used to back up the primary Astra Control Center instance or configure replication to a different cluster (in this example, the `ocp-cluster-3` cluster).
- **A third Kubernetes cluster of the same Kubernetes distribution type as the primary that will be used for restore operations**:
  - OpenShift cluster: `ocp-cluster-3`
  - Astra Control Center third instance: `ocp-cluster-3.company.com`
  - This cluster will be used for Astra Control Center restore or replication failover.

Ideally, the application cluster should be situated outside of the three Astra Control Center clusters as depicted by the kubernetes and rancher clusters in the image above.

Not depicted in the diagram:

- All the clusters have ONTAP backends with Astra Trident or Astra Control Provisioner installed.
- In this configuration, the OpenShift clusters are using MetalLB as the load balancer.
- The snapshot controller and VolumeSnapshotClass are also installed on all the clusters as outlined in the prerequisites.

Step 1 option: Backup and restore Astra Control Center

This procedure describes the necessary steps to restore Astra Control Center to a new cluster using backup and restore.

In this example, Astra Control Center is always installed under the `netapp-acc` namespace and the operator is installed under the `netapp-acc-operator` namespace.

Although not described, Astra Control Center operator can also be deployed in the same namespace as the Astra CR.

Before you begin

- You have installed the primary Astra Control Center on a cluster.
- You have installed the secondary Astra Control Center on a different cluster.

Steps

1. Manage the primary Astra Control Center application and destination cluster from the secondary Astra Control Center instance (running on `ocp-cluster-2` cluster):
   a. Log into the secondary Astra Control Center instance.
   b. Add the primary Astra Control Center cluster (`ocp-cluster-1`).
   c. Add the destination third cluster (`ocp-cluster-3`) that will be used for the restore.
2. Manage Astra Control Center and the Astra Control Center operator on the secondary Astra Control Center:
   a. From the Applications page, select **Define**.
   b. In the **Define application** window, enter the new application name (`netapp-acc`).
   c. Choose the cluster that is running the primary Astra Control Center (`ocp-cluster-1`) from the **Cluster** drop-down list.
   d. Choose the `netapp-acc` namespace for Astra Control Center from the **Namespace** drop-down list.
   e. On the Cluster Resources page, check **Include additional cluster-scoped resources**.
   f. Select **Add include rule**.
   g. Select these entries, and select **Add**:
      - Label selector: <label name>
      - Group: apiextensions.k8s.io
      - Version: v1

- Kind: CustomResourceDefinition

h. Confirm the application information.

i. Select **Define**.

After you select **Define**, repeat the Define Application process for the operator (`netapp-acc-operator`) and select the `netapp-acc-operator` namespace in the Define Application wizard.

3. Back up Astra Control Center and the operator:
   a. On the secondary Astra Control Center, navigate to the Applications page by selecting the Applications tab.
   b. Back up the Astra Control Center application (`netapp-acc`).
   c. Back up the operator (`netapp-acc-operator`).
4. After you have backed up Astra Control Center and the operator, simulate a disaster recovery (DR) scenario by uninstalling Astra Control Center from the primary cluster.

> You'll restore Astra Control Center to a new cluster (the third Kubernetes cluster described in this procedure) and use the same DNS as the primary cluster for the newly installed Astra Control Center.

5. Using the secondary Astra Control Center, restore the primary instance of the Astra Control Center application from its backup:
   a. Select **Applications** and then select the name of the Astra Control Center application.
   b. From the Options menu in the Actions column, select **Restore**.
   c. Choose the **Restore to new namespaces** as the restore type.
   d. Enter the restore name (`netapp-acc`).
   e. Choose the destination third cluster (`ocp-cluster-3`).
   f. Update the destination namespace so that it is the same namespace as the original.
   g. On the Restore Source page, select the application backup that will be used as the restore source.
   h. Select **Restore using original storage classes**.
   i. Select **Restore all resources**.
   j. Review restore information, and then select **Restore** to start the restore process that restores Astra Control Center to the destination cluster (`ocp-cluster-3`). The restore is complete when the application enters `available` state.
6. Configure Astra Control Center on the destination cluster:
   a. Open a terminal and connect using kubeconfig to the destination cluster (`ocp-cluster-3`) that contains the restored Astra Control Center.
   b. Confirm that the `ADDRESS` column in the Astra Control Center configuration references the primary system's DNS name:

```
kubectl get acc -n netapp-acc
```

Response:

```
NAME  UUID                                   VERSION    ADDRESS
READY
astra 89f4fd47-0cf0-4c7a-a44e-43353dc96ba8 24.02.0-69 ocp-cluster-
1.company.com           True
```

c. If the `ADDRESS` field in the above response does not have the FQDN of the primary Astra Control Center instance, update the configuration to reference the Astra Control Center DNS:

```
kubectl edit acc -n netapp-acc
```

i. Change the `astraAddress` under `spec:` to the FQDN (`ocp-cluster-1.company.com` in this example) of the primary Astra Control Center instance.

ii. Save the configuration.

iii. Confirm that the address has been updated:

```
kubectl get acc -n netapp-acc
```

d. Go to the Restore the Astra Control Center Operator section of this document to complete the restore process.

Step 1 option: Protect Astra Control Center using Replication

This procedure describes the necessary steps to configure Astra Control Center replication to protect the primary Astra Control Center instance.

In this example, Astra Control Center is always installed under the `netapp-acc` namespace and the operator is installed under the `netapp-acc-operator` namespace.

Before you begin

- You have installed the primary Astra Control Center on a cluster.
- You have installed the secondary Astra Control Center on a different cluster.

Steps

1. Manage the primary Astra Control Center application and destination cluster from the secondary Astra Control Center instance:
   a. Log into the secondary Astra Control Center instance.
   b. Add the primary Astra Control Center cluster (`ocp-cluster-1`).
   c. Add the destination third cluster (`ocp-cluster-3`) that will be used for the replication.
2. Manage Astra Control Center and the Astra Control Center operator on the secondary Astra Control Center:
   a. Select **Clusters** and select the cluster that contains the primary Astra Control Center (`ocp-cluster-1`).
   b. Select the **Namespaces** tab.

c. Select `netapp-acc` and `netapp-acc-operator` namespaces.

d. Select the Actions menu and select **Define as applications**.

e. Select **View in applications** to see the defined applications.

3. Configure Backends for Replication:

> Replication requires that the primary Astra Control Center cluster and the destination cluster (`ocp-cluster-3`) use different peered ONTAP storage backends.
> After each backend is peered and added to Astra Control, the backend appears in the **Discovered** tab of the Backends page.

a. Add a peered backend to Astra Control Center on the primary cluster.

b. Add a peered backend to Astra Control Center on the destination cluster.

4. Configure replication:

a. On the Applications screen, select the `netapp-acc` application.

b. Select **Configure replication policy**.

c. Select `ocp-cluster-3` as the destination cluster.

d. Select the storage class.

e. Enter `netapp-acc` as the destination namespace.

f. Change the replication frequency if desired.

g. Select **Next**.

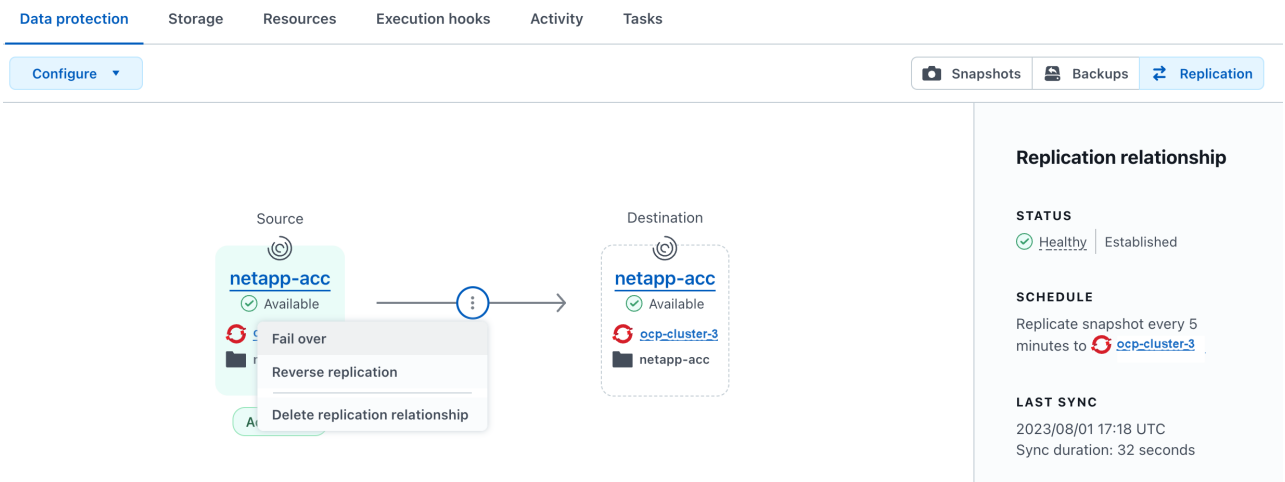
h. Confirm the configuration is correct, and select **Save**.

The replication relationship transitions from `Establishing` to `Established`. When active, this replication will occur every five minutes until the replication configuration is deleted.

5. Failover the replication to the other cluster if the primary system is corrupted or no longer accessible:

> Make sure the destination cluster does not have Astra Control Center installed to ensure a successful failover.

a. Select the vertical ellipses icon and select **Fail over**.



b. Confirm the details and select **Fail over** to begin the failover process.

   The replication relationship status changes to `Failing over` and then `Failed over` when complete.

6. Complete the failover configuration:
   a. Open a terminal and connect using the third cluster's kubeconfig (`ocp-cluster-3`). This cluster now has Astra Control Center installed.
   b. Determine the Astra Control Center FQDN on the third cluster (`ocp-cluster-3`).
   c. Update the configuration to reference the Astra Control Center DNS:

```
kubectl edit acc -n netapp-acc
```

      i. Change the `astraAddress` under `spec:` with the FQDN (`ocp-cluster-3.company.com`) of the destination third cluster.
      ii. Save the configuration.
      iii. Confirm that the address has been updated:

```
kubectl get acc -n netapp-acc
```

   d. Confirm that all required traefik CRDs are present:

```
kubectl get crds | grep traefik
```

   Required traefik CRDS:

```
ingressroutes.traefik.containo.us
ingressroutes.traefik.io
ingressroutetcps.traefik.containo.us
ingressroutetcps.traefik.io
ingressrouteudps.traefik.containo.us
ingressrouteudps.traefik.io
middlewares.traefik.containo.us
middlewares.traefik.io
middlewaretcps.traefik.containo.us
middlewaretcps.traefik.io
serverstransports.traefik.containo.us
serverstransports.traefik.io
tlsoptions.traefik.containo.us
tlsoptions.traefik.io
tIsstores.traefik.containo.us
tIsstores.traefik.io
traefikservices.traefik.containo.us
traefikservices.traefik.io
```

e. If some of the above CRDs are missing:

   i. Go to traefik documentation.

   ii. Copy the "Definitions" area into a file.

   iii. Apply changes:

```
kubectl apply -f <file name>
```

   iv. Restart traefik:

```
kubectl get pods -n netapp-acc | grep -e "traefik" | awk '{print
$1}' | xargs kubectl delete pod -n netapp-acc
```

f. Go to the Restore the Astra Control Center Operator section of this document to complete the restore process.

Step 2: Restore the Astra Control Center Operator

Using the secondary Astra Control Center, restore the primary Astra Control Center operator from backup. The destination namespace must be the same as the source namespace. In the case where Astra Control Center was deleted from the primary source cluster, backups will still exist to perform the same restore steps.

Steps

1. Select **Applications** and then select the name of the operator app (`netapp-acc-operator`).
2. From the Options menu in the Actions column, select **Restore**

3. Choose the **Restore to new namespaces** as the restore type.
4. Choose the destination third cluster (`ocp-cluster-3`).
5. Change the namespace to be the same as the namespace associated with the primary source cluster (`netapp-acc-operator`).
6. Select the backup that was taken earlier as the restore source.
7. Select **Restore using original storage classes**.
8. Select **Restore all resources**.
9. Review the details then click **Restore** to start the restore process.

   The Applications page shows the Astra Control Center operator being restored to the destination third cluster (`ocp-cluster-3`). When the process is complete, the state shows as `Available`. Within ten minutes, the DNS address should resolve on the page.

Result

Astra Control Center, its registered clusters, and managed applications with their snapshots and backups are now available on the destination third cluster (`ocp-cluster-3`). Any protection policies you had on the original are also there on the new instance. You can continue to take scheduled or on-demand backups and snapshots.

Troubleshooting

Determine system health and if protection processes were successful.

- **Pods are not running**: Confirm that all pods are up and running:

  ```
  kubectl get pods -n netapp-acc
  ```

  If some pods are in the `CrashLookBackOff` state, restart them and they should transition to `Running` state.

- **Confirm system status**: Confirm that the Astra Control Center system is in `ready` state:

  ```
  kubectl get acc -n netapp-acc
  ```

  Response:

  ```
  NAME  UUID                                   VERSION    ADDRESS
  READY
  astra 89f4fd47-0cf0-4c7a-a44e-43353dc96ba8 24.02.0-69 ocp-cluster-
  1.company.com           True
  ```

- **Confirm deployment status**: Show Astra Control Center deployment information to confirm that `Deployment State` is `Deployed`.

```
kubectl describe acc astra -n netapp-acc
```

- **Restored Astra Control Center UI returns a 404 error**: If this happens when you have selected `AccTraefik` as an ingress option, check the traefik CRDs to ensure they're all installed.

Monitor 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 cluster health and manage storage classes

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. You can also change the default storage class for managed clusters.

View cluster health and details

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.

Change the default storage class

You can change the default storage class for a cluster. When Astra Control manages a cluster, it keeps track of the cluster's default storage class.

> Do not change the storage class using kubectl commands. Use this procedure instead. Astra Control will revert the changes if made using kubectl.

Steps

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

View the health and details of an app

After you start managing an app, Astra Control provides details about the app that enables you to identify its communication status (whether Astra Control can communicate

with the app), 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. Review the information.

   App Status

   Provides a status that reflects whether Astra Control can communicate with the application.

   - **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 local users and roles

You can add, remove, and edit users of your Astra Control Center installation using the Astra Control UI. You can use the Astra Control UI or Astra Control API to manage users.

You can also use LDAP to perform authentication for selected users.

Use LDAP

LDAP is an industry standard protocol for accessing distributed directory information and a popular choice for

enterprise authentication. You can connect Astra Control Center to an LDAP server to perform authentication for selected Astra Control users. At a high level, the configuration involves integrating Astra with LDAP and defining the Astra Control users and groups corresponding to the LDAP definitions. You can use the Astra Control API or web UI to configure LDAP authentication and LDAP users and groups. See the following documentation for more information:

- Use the Astra Control API to manage remote authentication and users
- Use the Astra Control UI to manage remote users and groups
- Use the Astra Control UI to manage remote authentication

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, refer to Manage local users and 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**.

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 Astra Control API to manage roles.

Add a namespace constraint to a role

An Admin or Owner user can add namespace constraints to Member or Viewer roles.

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

Manage remote authentication

LDAP is an industry standard protocol for accessing distributed directory information and a popular choice for enterprise authentication. You can connect Astra Control Center to an LDAP server to perform authentication for selected Astra Control users.

At a high level, the configuration involves integrating Astra with LDAP and defining the Astra Control users and groups corresponding to the LDAP definitions. You can use the Astra Control API or web UI to configure LDAP authentication and LDAP users and groups.

> Astra Control Center uses the user login attribute, configured when remote authentication is enabled, to search for and keep track of remote users. An attribute of an email address ("mail") or user principal name ("userPrincipalName") must exist in this field for any remote user you wish to appear in Astra Control Center. This attribute is used as the username in Astra Control Center for authentication and in searches for remote users.

Add a certificate for LDAPS authentication

Add the private TLS certificate for the LDAP server so that Astra Control Center can authenticate with the LDAP server when you use an LDAPS connection. You only need to do this once, or when the certificate you have installed expires.

Steps

1. Go to **Account**.
2. Select the **Certificates** tab.
3. Select **Add**.
4. Either upload the `.pem` file or paste the contents of the file from your clipboard.
5. Select the **Trusted** check box.
6. Select **Add certificate**.

Enable remote authentication

You can enable LDAP authentication and configure the connection between Astra Control and the remote LDAP server.

Before you begin

If you plan to use LDAPS, ensure that the private TLS certificate for the LDAP server is installed in Astra Control Center so that Astra Control Center can authenticate with the LDAP server. See Add a certificate for LDAPS authentication for instructions.

Steps

1. Go to **Account > Connections**.
2. In the **Remote Authentication** pane, select the configuration menu.
3. Select **Connect**.
4. Enter the server IP address, port, and preferred connection protocol (LDAP or LDAPS).

As a best practice, use LDAPS when connecting with the LDAP server. You need to install the LDAP server's private TLS certificate in Astra Control Center before you connect with LDAPS.

5. Enter the service account credentials in email format (administrator@example.com). Astra Control will use these credentials when connecting with the LDAP server.
6. In the **User Match** section, do the following:
   a. Enter the base DN and an appropriate user search filter to use when retrieving user information from the LDAP server.
   b. (Optional) If your directory uses the user login attribute `userPrincipalName` instead of `mail`, enter `userPrincipalName` in the correct attribute in the **User login attribute** field.
7. In the **Group Match** section, enter the group search base DN and an appropriate custom group search filter.

Be sure to use the correct base Distinguished Name (DN) and an appropriate search filter for **User Match** and **Group Match**. The base DN tells Astra Control at what level of the directory tree to start the search, and the search filter limits the parts of the directory tree Astra Control searches from.

8. Select **Submit**.

Result

The **Remote Authentication** pane status moves to **Pending**, and then to **Connected** when the connection to the LDAP server is established.

Disable remote authentication

You can temporarily disable an active connection to the LDAP server.

When you disable a connection to an LDAP server, all settings are saved, and all remote users and groups that were added to Astra Control from that LDAP server are retained. You can reconnect to this LDAP server at any time.

Steps

1. Go to **Account > Connections**.
2. In the **Remote Authentication** pane, select the configuration menu.
3. Select **Disable**.

Result

The **Remote Authentication** pane status moves to **Disabled**. All remote authentication settings, remote users, and remote groups are preserved, and you can re-enable the connection at any time.

Edit remote authentication settings

If you have disabled the connection to the LDAP server or the **Remote Authentication** pane is in a "Connection error" state, you can edit the configuration settings.

You cannot edit the LDAP server URL or IP address when the **Remote Authentication** pane is in a "Disabled" state. You need to Disconnect remote authentication first.

Steps

1. Go to **Account > Connections**.
2. In the **Remote Authentication** pane, select the configuration menu.
3. Select **Edit**.
4. Make the necessary changes, and select **Edit**.

Disconnect remote authentication

You can disconnect from an LDAP server and remove the configuration settings from Astra Control.

> If you are an LDAP user and you disconnect, your session will immediately end. When you disconnect from the LDAP server, all configuration settings for that LDAP server are removed from Astra Control, as well as any remote users and groups that were added from that LDAP server.

Steps

1. Go to **Account > Connections**.
2. In the **Remote Authentication** pane, select the configuration menu.
3. Select **Disconnect**.

Result

The **Remote Authentication** pane status moves to **Disconnected**. Remote authentication settings, remote users, and remote groups are removed from Astra Control.

Manage remote users and groups

If you have enabled LDAP authentication on your Astra Control system, you can search for LDAP users and groups, and include them in the approved users of the system.

Add a remote user

Account Owners and Admins can add remote users to Astra Control. Astra Control Center supports up to 10,000 LDAP remote users.

> Astra Control Center uses the user login attribute, configured when remote authentication is enabled, to search for and keep track of remote users. An attribute of an email address ("mail") or user principal name ("userPrincipalName") must exist in this field for any remote user you wish to appear in Astra Control Center. This attribute is used as the username in Astra Control Center for authentication and in searches for remote users.

> You cannot add a remote user if a local user with the same email address (based on the "mail" or "user principal name" attribute) already exists on the system. To add the user as a remote user, delete the local user from the system first.

Steps

1. Go to the **Account** area.
2. Select the **Users & groups** tab.
3. At the far right of the page, select **Remote users**.

4. Select **Add**.
5. Optionally, search for an LDAP user by entering the user's email address in the **Filter by email** field.
6. Select one or more users from the list.
7. Assign a role to the user.

   If you assign different roles to a user and the user's group, the more permissive role takes precedence.

8. Optionally, assign one or more namespace constraints to this user, and select **Restrict role to constraints** to enforce them. You can add a new namespace constraint by selecting **Add constraint**.

   When a user is assigned multiple roles through LDAP group membership, the constraints in the most permissive role are the only ones that take effect. For example, if a user with a local Viewer role joins three groups that are bound to the Member role, the sum of the constraints from the Member roles take effect, and any constraints from the Viewer role are ignored.

9. Select **Add**.

Result

The new user appears in the list of remote users. In this list, you can see active constraints on the user as well as manage the user from the **Actions** menu.

Add a remote group

To add many remote users at once, account Owners and Admins can add remote groups to Astra Control. When you add a remote group, all remote users in that group are available to log in to Astra Control and will inherit the same role as the group.

Astra Control Center supports up to 5,000 LDAP remote groups.

Steps

1. Go to the **Account** area.
2. Select the **Users & groups** tab.
3. At the far right of the page, select **Remote groups**.
4. Select **Add**.

   In this window, you can see a list of the common names and distinguished names of LDAP groups that Astra Control retrieved from the directory.

5. Optionally, search for an LDAP group by entering the group's common name in the **Filter by common name** field.
6. Select one or more groups from the list.
7. Assign a role to the groups.

   The role you select is assigned to all users in this group. If you assign different roles to a user and the user's group, the more permissive role takes precedence.

8. Optionally, assign one or more namespace constraints to this group, and select **Restrict role to**

constraints to enforce them. You can add a new namespace constraint by selecting **Add constraint**.

- **If the resources being accessed belong to clusters that have the latest Astra Connector installed**: When a user is assigned multiple roles through LDAP group membership, the constraints from the roles are combined. For example, if a user with a local Viewer role joins three groups that are bound to the Member role, the user now has Viewer role access to the original resources as well as Member role access to the resources gained through group membership.
- **If the resources being accessed belong to clusters that do not have Astra Connector installed**: When a user is assigned multiple roles through LDAP group membership, the constraints from the most permissive role are the only ones that take effect.

9. Select **Add**.

Result

The new group appears in the list of remote groups. Remote users in this group don't appear in the list of remote users until each remote user logs in. In this list, you can see details about the group as well as manage the group from the **Actions** menu.

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, refer to Add a Kubernetes cluster.

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

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 Control Center UI or 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 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

- Microsoft Azure
- Generic S3

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

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

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
- Set the default bucket
- Rotate or remove bucket credentials
- Remove a bucket
- [Tech preview] Manage a bucket using a custom resource

> 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 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**.

Set the default bucket

When you perform a clone across clusters, Astra Control requires a default bucket. Follow these steps to set a default bucket for all clusters.

Steps

1. Go to **Cloud instances**.
2. Select the menu in the **Actions** column for the cloud instance in the list.
3. Select **Edit**.
4. In the **Bucket** list, select the bucket you want to be the default.
5. Select **Save**.

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.
- You cannot remove a write once read many (WORM) bucket before the bucket's cloud provider retention period has expired. WORM buckets are denoted with "Locked" next to the bucket name.

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

Before you begin

- 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'll 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**.

[Tech preview] Manage a bucket using a custom resource

You can add a bucket using the an Astra Control custom resource (CR) on the application cluster. 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. If you are using the custom resource method, application snapshots functionality requires a bucket.

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

The bucket custom resource for Astra Control is known as an AppVault. This CR contains the configurations necessary for a bucket to be used in protection operations.

Before you begin

- Ensure you have a bucket that is reachable from your clusters managed by Astra Control Center.
- Ensure you have credentials for the bucket.
- Ensure the bucket is one of the following types:

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

> Amazon Web Services (AWS) uses the Generic S3 bucket type.

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

Steps

1. Create the custom resource (CR) file and name it (for example, `astra-appvault.yaml`).
2. Configure the following attributes:
   - **metadata.name**: *(Required)* The name of the AppVault custom resource.
   - **spec.prefix**: *(Optional)* A path that is prefixed to the names of all entities stored in the AppVault.
   - **spec.providerConfig**: *(Required)* Stores the configuration necessary to access the AppVault using the specified provider.
   - **spec.providerCredentials**: *(Required)* Stores references to any credential required to access the AppVault using the specified provider.
     - **spec.providerCredentials.valueFromSecret**: *(Optional)* Indicates that the credential value should come from a secret.
       - **key**: *(Required if valueFromSecret is used)* The valid key of the secret to select from.
       - **name**: *(Required if valueFromSecret is used)* Name of the secret containing the value for this field. Must be in the same namespace.
   - **spec.providerType**: *(Required)* Determines what provides the backup; for example, NetApp ONTAP S3 or Microsoft Azure.

     Example YAML:

```
apiVersion: astra.netapp.io/v1
kind: AppVault
metadata:
  name: astra-appvault
spec:
  providerType: generic-s3
  providerConfig:
    path: testpath
    endpoint: 192.168.1.100:80
    bucketName: bucket1
    secure: "false"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        name: s3-creds
        key: accessKeyID
    secretAccessKey:
      valueFromSecret:
        name: s3-creds
        key: secretAccessKey
```

3. After you populate the `astra-appvault.yaml` file with the correct values, apply the CR:

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

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

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.

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
- Edit storage backend authentication details
- Manage a discovered storage backend
- Unmanage a storage backend
- 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 panel of the Dashboard 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).

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.

Edit storage backend authentication details

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

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

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

For details on enabling certificate-based authentication, refer to Enable authentication on the ONTAP storage backend.

Steps

1. From the left navigation, select **Backends**.

2. Select the storage backend.
3. At the Credentials field, select the **Edit** icon.
4. In the Edit page, select one of the following.
   - **Use administrator credentials**: Enter the ONTAP cluster management IP address and admin credentials. The credentials must be cluster-wide credentials.

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

   - **Use a certificate**: Upload the certificate `.pem` file, the certificate key `.key` file, and optionally the certificate authority file.
5. Select **Save**.

Manage a discovered storage backend

You can select to manage an unmanaged, yet discovered storage backend. When you manage a storage backend, Astra Control indicates if a certificate for authentication has expired.

Steps

1. From the left navigation, select **Backends**.
2. Select the **Discovered** option.
3. Select the storage backend.
4. From the Options menu in the **Actions** column, select **Manage**.
5. Make the changes.
6. Select **Save**.

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.

Before you begin

- Ensure that the storage backend is unmanaged.
- Ensure that the storage backend does not have any volumes associated with the 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**.

Find more information

- Use the Astra Control API

Monitor running tasks

You can view details about running tasks and tasks that have completed, failed, or been cancelled in the last 24 hours in Astra Control. For example, you can view the status of a running backup, restore, or clone operation, and see details like percentage completed and estimated time remaining. You can view the status of a scheduled operation that has run or an operation that you started manually.

While viewing a running or completed task, you can expand the task details to see the status of each of the subtasks. The task progress bar is green for ongoing or completed tasks, blue for cancelled tasks, and red for tasks that failed because of an error.

For clone operations, the task subtasks consist of a snapshot and a snapshot restore operation.

To see more information about failed tasks, refer to Monitor account activity.

Steps

1. While a task is running, go to **Applications**.
2. Select the name of an application from the list.
3. In the details of the application, select the **Tasks** tab.

   You can view details of current or past tasks, and filter by task state.

Tasks are retained in the **Tasks** list for up to 24 hours. You can configure this limit and other task monitor settings using the Astra Control API.

[Tech preview] Manage Astra Control applications using CRs

Manage your Astra Control applications using Kubernetes custom resources (CR). The following options are available:

- Define an application using a Kubernetes custom resource
- Manage a bucket using a custom resource

Monitor infrastructure with Prometheus or Fluentd connections

You can configure several optional settings to enhance your Astra Control Center experience. To monitor and gain insight into your complete infrastructure, configure Prometheus or add a Fluentd connection.

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

- Connect to Prometheus
- Connect to Fluentd

Add a proxy server for connections to the NetApp Support Site

If the network where you're running Astra Control Center requires a proxy for connecting to the Internet (to upload support bundles to the NetApp Support Site), 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 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.

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.

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'll 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 Prometheus

You can monitor Astra Control Center data with Prometheus. You can configure Prometheus to gather metrics from the Kubernetes cluster metrics endpoint, and you can use Prometheus also to visualize the metrics data.

For details about using Prometheus, refer to their documentation at Getting started with Prometheus.

What you'll need

Make sure that you have downloaded and installed the Prometheus package on the Astra Control Center

cluster or a different cluster that can communicate with the Astra Control Center cluster.

Follow the instructions in the official documentation to Install Prometheus.

Prometheus needs to be able to communicate with the Astra Control Center Kubernetes cluster. If Prometheus is not installed on the Astra Control Center cluster, you need to make sure they can communicate with the metrics service running on the Astra Control Center cluster.

Configure Prometheus

Astra Control Center exposes a metrics service on TCP port 9090 in the Kubernetes cluster. You need to configure Prometheus to collect metrics from this service.

Steps

1. Log into the Prometheus server.
2. Add your cluster entry into the `prometheus.yml` file. In the `yml` file, add an entry similar to the following for your cluster in the `scrape_configs section`:

```
job_name: '<Add your cluster name here. You can abbreviate. It just
needs to be a unique name>'
  metrics_path: /accounts/<replace with your account ID>/metrics
  authorization:
     credentials: <replace with your API token>
  tls_config:
     insecure_skip_verify: true
  static_configs:
    - targets: ['<replace with your astraAddress. If using FQDN, the
prometheus server has to be able to resolve it>']
```

> If you set the `tls_config insecure_skip_verify` to `true`, the TLS encryption protocol is not required.

3. Restart the Prometheus service:

```
sudo systemctl restart prometheus
```

Access Prometheus

Access the Prometheus URL.

Steps

1. In a browser, enter the Prometheus URL with port 9090.
2. Verify your connection by selecting **Status** > **Targets**.

View data in Prometheus

You can use Prometheus to view Astra Control Center data.

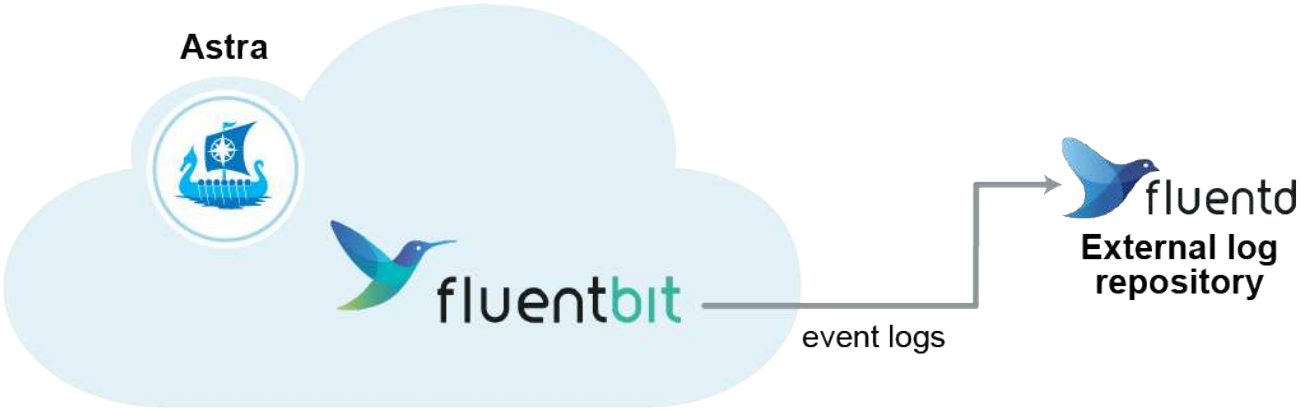
Steps

1. In a browser, enter the Prometheus URL.
2. From the Prometheus menu, select **Graph**.
3. To use the Metrics Explorer, select the icon next to **Execute**.
4. Select `scrape_samples_scraped` and select **Execute**.
5. To see sample scraping over time, select **Graph**.

If multiple cluster data was collected, each cluster's metrics appear in a different color.

Connect to Fluentd

You can send logs (Kubernetes events) from a system monitored by 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.

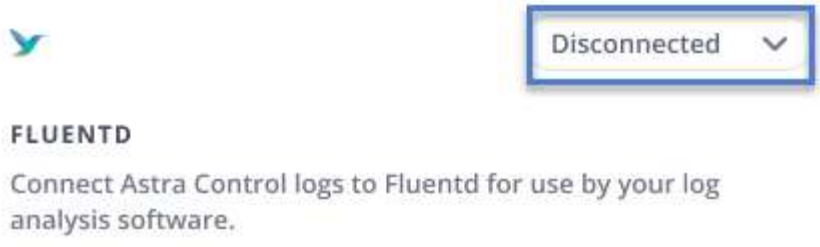
Before you begin

- 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.

When you unmanage an app:

- 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 app.
3. From the Options menu in the Actions column, select **Unmanage**.
4. Review the information.
5. Type "unmanage" to confirm.
6. Select **Yes, unmanage application**.

Result

Astra Control Center stops managing the app.

Unmanage a cluster

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

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

When you unmanage a cluster:

- 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.
- Astra Control Provisioner or Astra Trident won't be uninstalled from the cluster. Learn how to uninstall Astra Trident.

Steps

1. From the left navigation bar, select **Clusters**.
2. Select the check box for the cluster that you no longer want to manage.
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**. After that, the cluster will be removed from the **Clusters** page and it is no longer managed by Astra Control Center.

Unmanaging the cluster removes all the resources that were installed for sending telemetry data.

Upgrade Astra Control Center

To upgrade Astra Control Center, download the installation images and complete these instructions. You can use this procedure to upgrade Astra Control Center in internet-connected or air-gapped environments.

These instructions describe the upgrade process for Astra Control Center from the second-most recent release to this current release. You cannot upgrade directly from a version that is two or more releases behind the current release. If your installed Astra Control Center version is many versions behind the latest release, you might need to perform chain upgrades to more recent versions until your installed Astra Control Center is only one version behind the latest release. For a complete list of released versions, see the release notes.

Before you begin

Before you upgrade, ensure your environment still meets the minimum requirements for Astra Control Center deployment. Your environment should have the following:

- **An enabled Astra Control Provisioner with Astra Trident running**
  1. Determine the Astra Trident version you are running:

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

     If you are running Astra Trident 23.01 or earlier, use these instructions to upgrade to a more recent version of Astra Trident before upgrading to the Astra Control Provisioner. You can perform a direct upgrade to Astra Control Provisioner 24.02 if your Astra Trident is within a four-release window of version 24.02. For example, you can directly upgrade from Astra Trident 23.04 to Astra Control Provisioner 24.02.

  2. Verify that Astra Control Provisioner has been enabled. Astra Control Provisioner will not work with releases of Astra Control Center earlier than 23.10. Upgrade your Astra Control Provisioner so that it has the same version as the Astra Control Center you are upgrading to access the latest functionality.
- **A supported Kubernetes distribution**

  Determine the Kubernetes version you are running:

  ```
  kubectl get nodes -o wide
  ```

- **Sufficient cluster resources**

  Determine available cluster resources:

  ```
  kubectl describe node <node name>
  ```

- **A default storage class**

  Determine your default storage class:

  ```
  kubectl get storageclass
  ```

- **Healthy and available API services**

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

  ```
  kubectl get apiservices
  ```

- **(Local registries only) A local registry you can use to push and upload Astra Control Center images**
- **(OpenShift only) Healthy and available cluster operators**

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

  ```
  kubectl get clusteroperators
  ```

You should also consider the following:

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

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

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

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

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

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

- **Istio service mesh deployments**
  If you installed an Istio service mesh during Astra Control Center installation, this upgrade of Astra Control Center will include Istio service mesh. If you do not yet have a service mesh, you can only install one during an initial deployment of Astra Control Center.

About this task

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

Log out of your Astra Control Center UI before you begin the upgrade.

- Download and extract Astra Control Center
- Complete additional steps if you use a local registry
- Install the updated Astra Control Center operator
- Upgrade Astra Control Center
- Verify system status

Do not delete the Astra Control Center operator (for example, `kubectl delete -f astra_control_center_operator_deploy.yaml`) at any time during the Astra Control Center upgrade or operation to avoid deleting pods.

Download and extract Astra Control Center

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

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

Astra Control image registry

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

NetApp Support Site

1. Download the bundle containing Astra Control Center (`astra-control-center-[version].tar.gz`) from the Astra Control Center downloads page.
2. (Recommended but optional) Download the certificates and signatures bundle for Astra Control Center (`astra-control-center-certs-[version].tar.gz`) to verify the signature of the bundle.

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

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

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

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

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

Complete additional steps if you use a local registry

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

Remove the NetApp Astra kubectl plugin and install it again

You need to use the latest version of the NetApp Astra kubectl command line plugin to push images to a local Docker repository.

1. Determine if you have the plug-in installed:

   ```
   kubectl astra
   ```

2. Take one of these actions:
   - If the plugin is installed, the command should return the kubectl plugin help and you can remove the existing version of kubectl-astra: `delete /usr/local/bin/kubectl-astra`.

- If the command returns an error, the plugin is not installed and you can proceed to the next step to install it.

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

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

      ```
      ls kubectl-astra/
      ```

   b. Move the correct binary into the current path and rename it to `kubectl-astra`:

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

Add the images to your registry

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

Docker

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

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

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

- Replace <BUNDLE_FILE> with the name of the Astra Control bundle file (`acc.manifest.bundle.yaml`).
- Replace <MY_FULL_REGISTRY_PATH> with the URL of the Docker repository; for example, "https://<docker-registry>".
- Replace <MY_REGISTRY_USER> with the user name.
- Replace <MY_REGISTRY_TOKEN> with an authorized token for the registry.

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

Podman

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

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

b. Log in to your registry:

```
podman login <YOUR_REGISTRY>
```

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

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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

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

2. Change the directory:

```
cd manifests
```

Install the updated Astra Control Center operator

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

   a. Open the Astra Control Center operator deployment YAML:

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   An annotated sample YAML follows these steps.

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

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

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

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

   e. Add the following values to the `env` section:

   ```
   - name: ACCOP_HELM_UPGRADETIMEOUT
     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
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
```

```
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: ASTRA_IMAGE_REGISTRY/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_UPGRADETIMEOUT
          value: 300m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
            memory: 75Mi
        securityContext:
          allowPrivilegeEscalation: false
      imagePullSecrets: []
      securityContext:
```

```
        runAsUser: 65532
      terminationGracePeriodSeconds: 10
```

2. Install the 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.as
tra.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
```

3. Verify pods are running:

```
kubectl get pods -n netapp-acc-operator
```

Upgrade Astra Control Center

1. Edit the Astra Control Center custom resource (CR):

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

An annotated sample YAML follows these steps.

2. Change the Astra version number (`astraVersion` inside of `spec`) from `23.10.0` to `24.02.0`:

   You cannot upgrade directly from a version that is two or more releases behind the current release. For a complete list of released versions, see the release notes.

```
spec:
  accountName: "Example"
  astraVersion: "[Version number]"
```

3. Change the image registry:
   - (Local registries only) If you are using a local registry, verify that your image registry path matches the registry path you pushed the images to in a previous step. Update `imageRegistry` inside of `spec` if the local registry has changed since your last installation.
   - (Astra Control image registry) Use the Astra Control image registry (`cr.astra.netapp.io`) you used to download the updated Astra Control bundle.

```
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
```

4. Add the following to your `crds` configuration inside of `spec`:

```
crds:
  shouldUpgrade: true
```

5. Add the following lines within `additionalValues` inside of `spec` in the Astra Control Center CR:

```
additionalValues:
    nautilus:
      startupProbe:
        periodSeconds: 30
        failureThreshold: 600
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

6. Save and exit the file editor. The changes will be applied and the upgrade will begin.
7. (Optional) Verify that the pods terminate and become available again:

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

8. Wait for the Astra Control status conditions to indicate that the upgrade is complete and ready (`True`):

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

Response:

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   24.02.0-69
10.111.111.111  True
```

To monitor upgrade status during the operation, run the following command: `kubectl get AstraControlCenter -o yaml -n [netapp-acc or custom namespace]`

To inspect the Astra Control Center operator logs, run the following command: `kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-operator -c manager -f`

Verify system status

1. Log in to Astra Control Center.
2. Verify that the version has been upgraded. See the **Support** page in the UI.
3. Verify that all your managed clusters and apps are still present and protected.

Upgrade Astra Control Center using OpenShift OperatorHub

If you installed Astra Control Center using its Red Hat-certified operator, you can upgrade Astra Control Center using an updated operator from OperatorHub. Use this procedure to upgrade Astra Control Center from the Red Hat Ecosystem Catalog or using the Red Hat OpenShift Container Platform.

Before you begin

- **Meet environmental prerequisites**: Before you upgrade, ensure your environment still meets the minimum requirements for Astra Control Center deployment.
- **Ensure that you have enabled Astra Control Provisioner with Astra Trident running**
  1. Determine the Astra Trident version you are running:

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

If you are running Astra Trident 23.01 or earlier, use these instructions to upgrade to a more recent version of Astra Trident before upgrading to the Astra Control Provisioner. You can perform a direct upgrade to Astra Control Provisioner 24.02 if your Astra Trident is within a four-release window of version 24.02. For example, you can directly upgrade from Astra Trident 23.04 to Astra Control Provisioner 24.02.

2. Verify that Astra Control Provisioner has been enabled. Astra Control Provisioner will not work with releases of Astra Control Center earlier than 23.10. Upgrade your Astra Control Provisioner so that it has the same version as the Astra Control Center you are upgrading to access the latest functionality.

- **Ensure healthy cluster operators and API services**:
  - From your OpenShift cluster, ensure all cluster operators are in a healthy state:

```
oc get clusteroperators
```

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

```
oc get apiservices
```

- **OpenShift permissions**: You have all necessary permissions and access to the Red Hat OpenShift Container Platform to perform the upgrade steps described.
- **(ONTAP SAN driver only) Enable multipath**: If you are using an ONTAP SAN driver, be sure that multipath is enabled on all your Kubernetes clusters.

You should also consider the following:

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

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

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

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

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

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

Steps

- Access the operator install page
- Uninstall the existing operator
- Install the latest operator

- Upgrade Astra Control Center

Access the operator install page

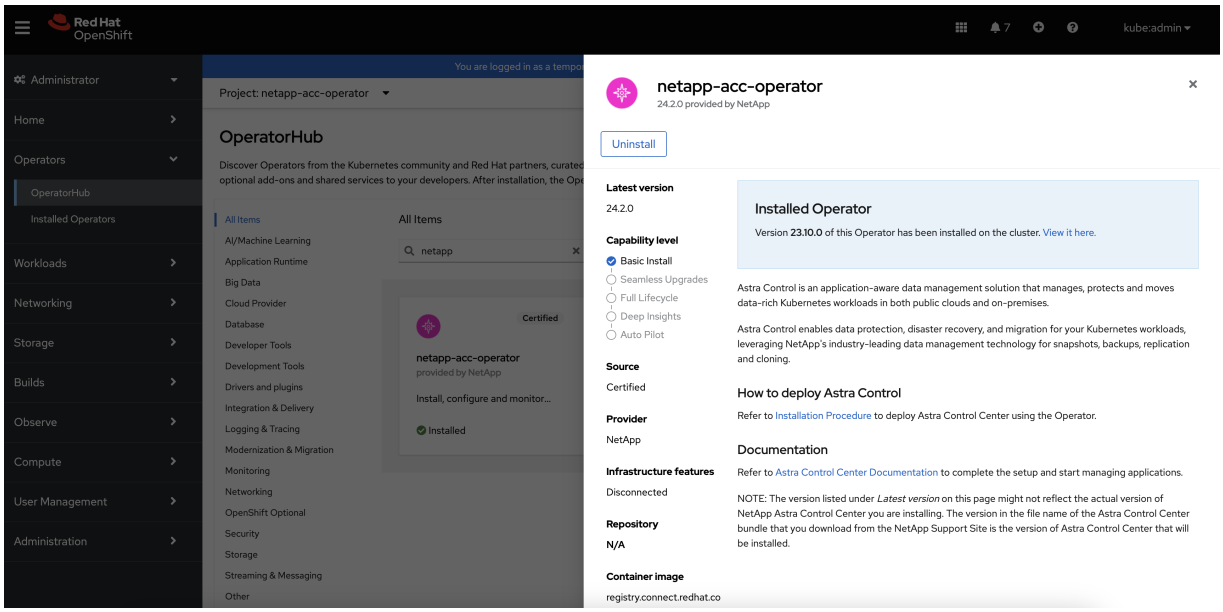
1. Complete the corresponding procedure for either Openshift Container Platform or Ecosystem Catalog:

Red Hat OpenShift web console

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

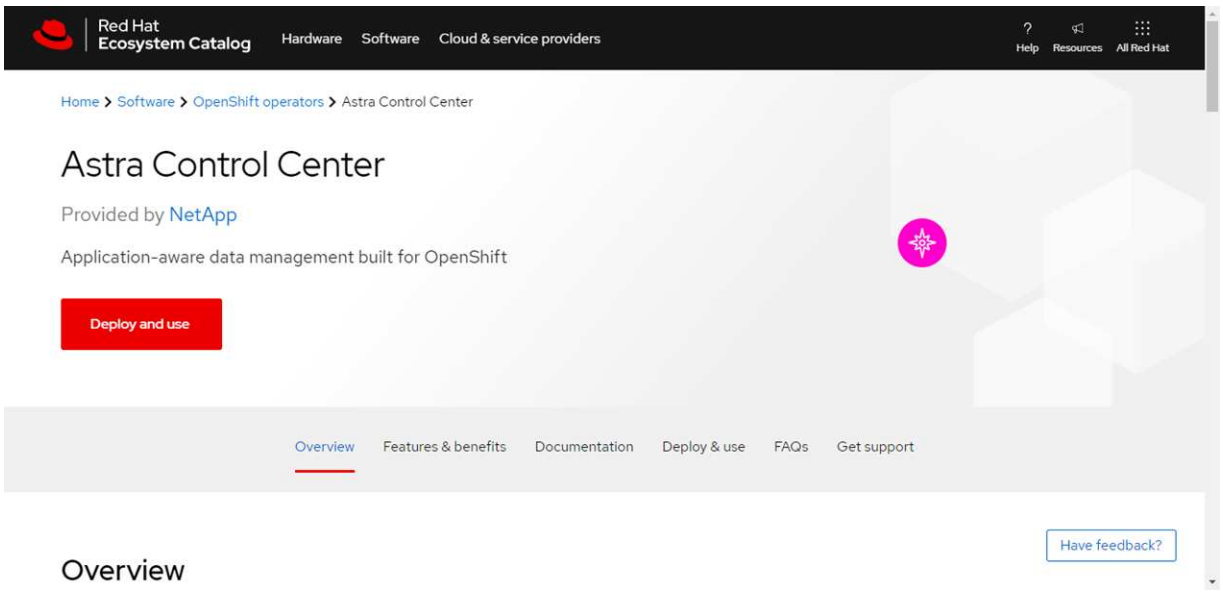
> You can upgrade only to the current version of Astra Control Center using this operator.

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



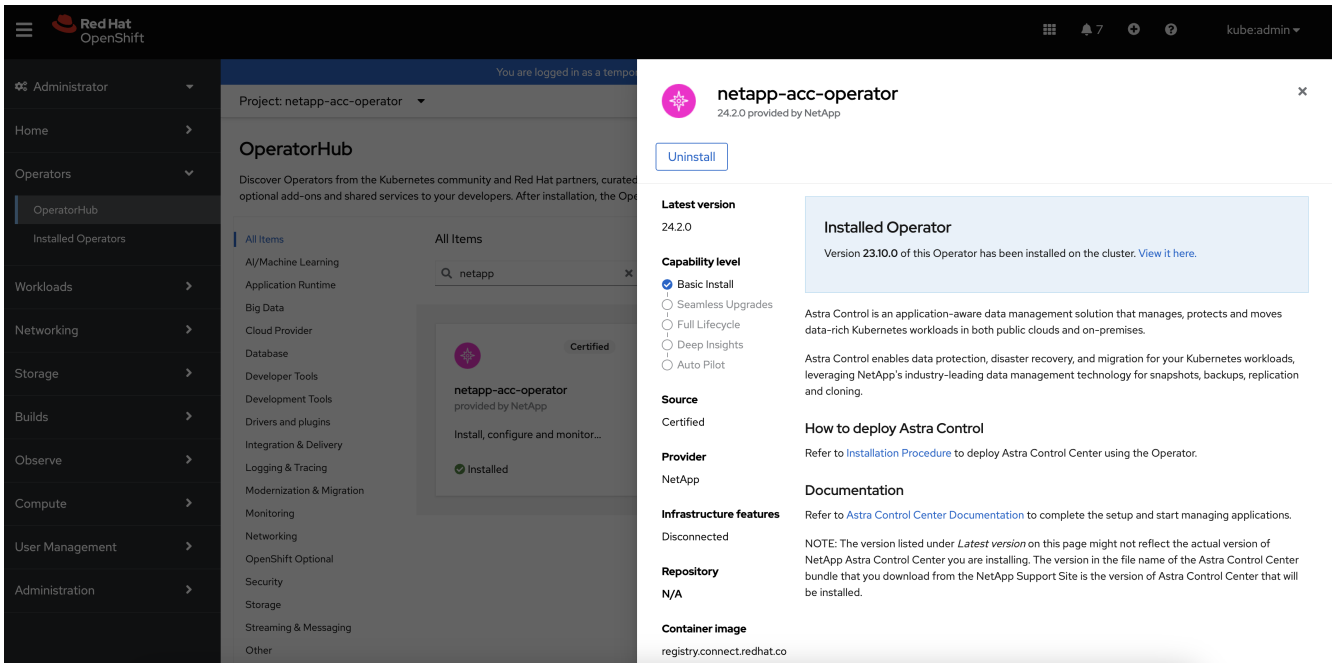
Red Hat Ecosystem Catalog

1. Select the NetApp Astra Control Center operator.
2. Select **Deploy and use**.



Uninstall the existing operator

1. From the **netapp-acc-operator** page, select **Uninstall** to remove your existing operator.

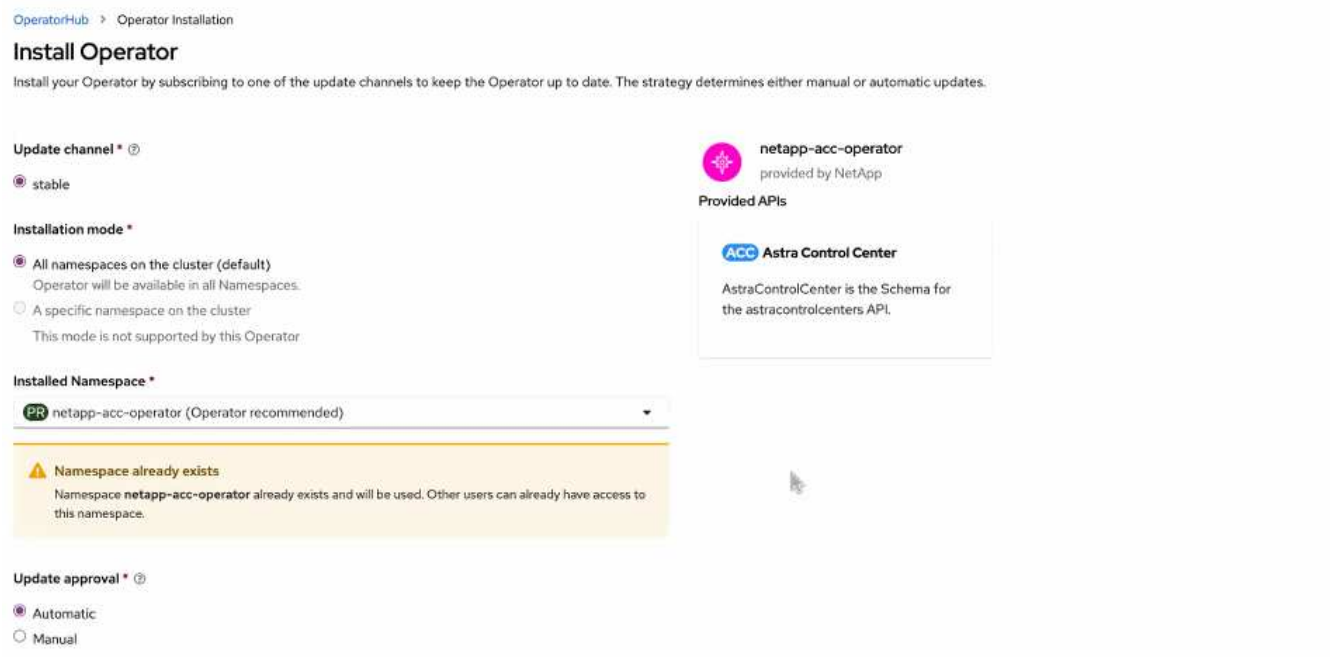


2. Confirm the operation.

> This operation deletes the netapp-acc-operator but preserves the original associated namespace and resources, such as secrets.

Install the latest operator

1. Navigate to the `netapp-acc` operator page again.
2. Complete the **Install Operator** page and install the most recent operator:



The operator will be available in all cluster namespaces.

a. Select the operator's `netapp-acc-operator` namespace (or custom namespace) that remains from the deleted operator's previous installation.

b. Select a manual or automatic approval strategy.

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

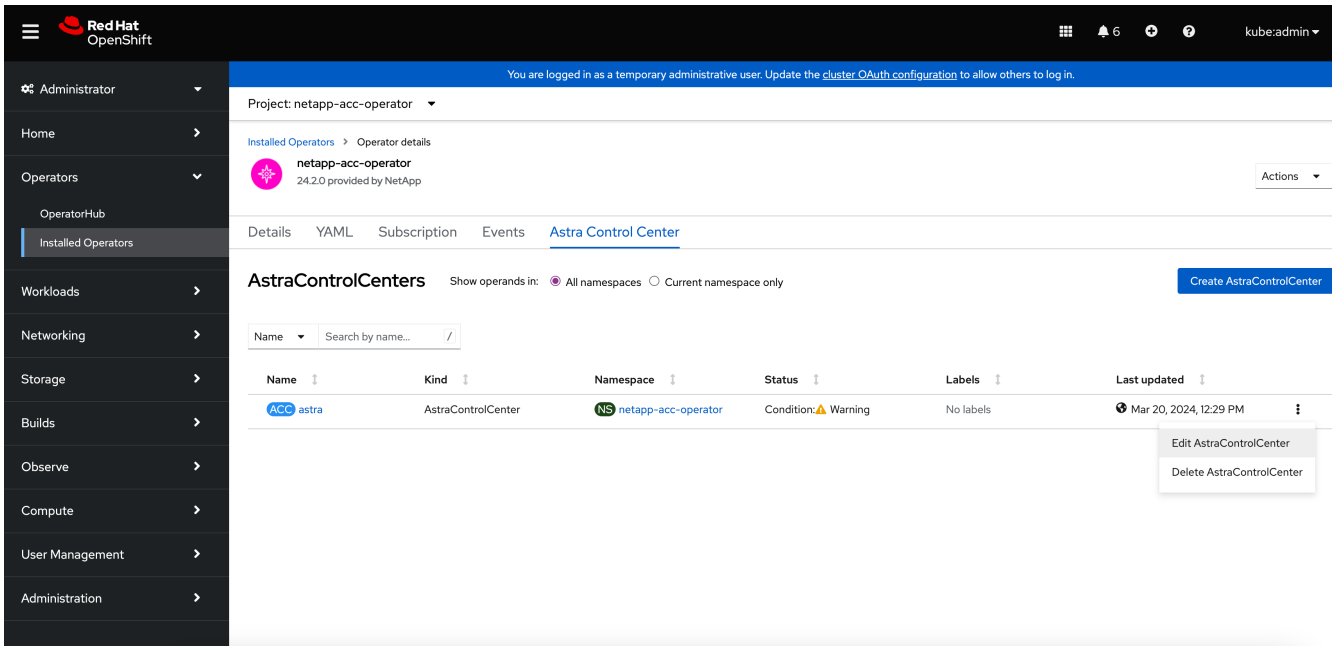
c. Select **Install**.

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

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

Upgrade Astra Control Center

1. From the Astra Control Center operator tab, select the Astra Control Center that remains from the previous installation and select **Edit AstraControlCenter**.



2. Update the `AstraControlCenter` YAML:
   a. Enter the latest Astra Control Center version; for example, 24.02.0-69.
   b. In `imageRegistry.name`, update the image registry path as needed:
      - If you are using the Astra Control registry option, change the path to `cr.astra.netapp.io`.
      - If you configured a local registry, change or retain the local image registry path where you pushed the images in a previous step.

      Do not enter `http://` or `https://` in the address field.

   c. Update the `imageRegistry.secret` as needed.

      The operator uninstall process does not remove existing secrets. You only need to update this field if you create a new secret with a different name from the existing secret.

   d. Add the following to your `crds` configuration:

```
crds:
  shouldUpgrade: true
```

3. Save your changes.
4. The UI confirms that the upgrade was successful.

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.

Before you begin

1. Unmanage all apps on the clusters.
2. 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.yaml -n netapp-acc
   ```

   Result:

   ```
   astracontrolcenter.astra.netapp.io "astra" deleted
   ```

2. Use the following command to delete the `netapp-acc` (or custom-named) namespace:

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

   Example 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

```
ingressroutes.traefik.containo.us              2021-06-23T23:29:11Z
ingressroutetcps.traefik.containo.us           2021-06-23T23:29:11Z
ingressrouteudps.traefik.containo.us           2021-06-23T23:29:12Z
middlewares.traefik.containo.us                2021-06-23T23:29:12Z
middlewaretcps.traefik.containo.us             2021-06-23T23:29:12Z
serverstransports.traefik.containo.us          2021-06-23T23:29:13Z
tlsoptions.traefik.containo.us                 2021-06-23T23:29:13Z
tlsstores.traefik.containo.us                  2021-06-23T23:29:14Z
traefikservices.traefik.containo.us            2021-06-23T23:29:15Z
```

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
middlewaretcps.traefik.containo.us
```

Find more information

- Known issues for uninstall

Use Astra Control Provisioner

Configure storage backend encryption

Using Astra Control Provisioner, you can improve data access security by enabling encryption for the traffic between your managed cluster and the storage backend.

Astra Control Provisioner supports Kerberos encryption for two types of storage backends:

- **On-premises ONTAP** - Astra Control Provisioner supports Kerberos encryption over NFSv3 and NFSv4 connections from Red Hat OpenShift and upstream Kubernetes clusters to on-premise ONTAP volumes.
- **Azure NetApp Files** - Astra Control Provisioner supports Kerberos encryption over NFSv4.1 connections from upstream Kubernetes clusters to Azure NetApp Files volumes.

You can create, delete, resize, snapshot, clone, read-only clone, and import volumes that use NFS encryption.

Configure in-flight Kerberos encryption with on-premises ONTAP volumes

You can enable Kerberos encryption on the storage traffic between your managed cluster and an on-premises ONTAP storage backend.

> Kerberos encryption for NFS traffic with on-premises ONTAP storage backends is only supported using the `ontap-nas` storage driver.

Before you begin

- Ensure that you have enabled Astra Control Provisioner on the managed cluster.
- Ensure that you have access to the `tridentctl` utility.
- Ensure you have administrator access to the ONTAP storage backend.
- Ensure you know the name of the volume or volumes you'll be sharing from the ONTAP storage backend.
- Ensure that you have prepared the ONTAP storage VM to support Kerberos encryption for NFS volumes. Refer to Enable Kerberos on a data LIF for instructions.
- Ensure that any NFSv4 volumes you use with Kerberos encryption are configured correctly. Refer to the NetApp NFSv4 Domain Configuration section (page 13) of the NetApp NFSv4 Enhancements and Best Practices Guide.

Add or modify ONTAP export policies

You need to add rules to existing ONTAP export policies or create new export polices that support Kerberos encryption for the ONTAP storage VM root volume as well as any ONTAP volumes shared with the upstream Kubernetes cluster. The export policy rules you add, or new export policies you create, need to support the following access protocols and access permissions:

Access protocols

Configure the export policy with NFS, NFSv3, and NFSv4 access protocols.

Access details

You can configure one of three different versions of Kerberos encryption, depending on your needs for the volume:

- **Kerberos 5** - (authentication and encryption)
- **Kerberos 5i** - (authentication and encryption with identity protection)
- **Kerberos 5p** - (authentication and encryption with identity and privacy protection)

Configure the ONTAP export policy rule with the appropriate access permissions. For example, if clusters will be mounting the NFS volumes with a mixture of Kerberos 5i and Kerberos 5p encryption, use the following access settings:

Type	Read-only access	Read/Write access	Superuser access
UNIX	Enabled	Enabled	Enabled
Kerberos 5i	Enabled	Enabled	Enabled
Kerberos 5p	Enabled	Enabled	Enabled

Refer to the following documentation for how to create ONTAP export policies and export policy rules:

- Create an export policy
- Add a rule to an export policy

Create a storage backend

You can create an Astra Control Provisioner storage backend configuration that includes Kerberos encryption capability.

About this task

When you create a storage backend configuration file that configures Kerberos encryption, you can specify one of three different versions of Kerberos encryption using the `spec.nfsMountOptions` parameter:

- `spec.nfsMountOptions: sec=krb5` (authentication and encryption)
- `spec.nfsMountOptions: sec=krb5i` (authentication and encryption with identity protection)
- `spec.nfsMountOptions: sec=krb5p` (authentication and encryption with identity and privacy protection)

Specify only one Kerberos level. If you specify more than one Kerberos encryption level in the parameter list, only the first option is used.

Steps

1. On the managed cluster, create a storage backend configuration file using the following example. Replace values in brackets <> with information from your environment:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Use the configuration file you created in the previous step to create the backend:

```
tridentctl create backend -f <backend-configuration-file>
```

If the backend creation fails, something is wrong with the backend configuration. You can view the logs to determine the cause by running the following command:

```
tridentctl logs
```

After you identify and correct the problem with the configuration file, you can run the create command again.

Create a storage class

You can create a storage class to provision volumes with Kerberos encryption.

About this task

When you create a storage class object, you can specify one of three different versions of Kerberos encryption using the `mountOptions` parameter:

- `mountOptions: sec=krb5` (authentication and encryption)
- `mountOptions: sec=krb5i` (authentication and encryption with identity protection)
- `mountOptions: sec=krb5p` (authentication and encryption with identity and privacy protection)

Specify only one Kerberos level. If you specify more than one Kerberos encryption level in the parameter list, only the first option is used. If the level of encryption you specified in the storage backend configuration is different than the level you specify in the storage class object, the storage class object takes precedence.

Steps

1. Create a StorageClass Kubernetes object, using the following example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
parameters:
  backendType: "ontap-nas"
  storagePools: "ontapnas_pool"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: True
```

2. Create the storage class:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Make sure that the storage class has been created:

```
kubectl get sc ontap-nas-sc
```

   You should see output similar to the following:

```
NAME            PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provision volumes

After you create a storage backend and a storage class, you can now provision a volume. Refer to these instructions for provisioning a volume.

Configure in-flight Kerberos encryption with Azure NetApp Files volumes

You can enable Kerberos encryption on the storage traffic between your managed cluster and a single Azure NetApp Files storage backend or a virtual pool of Azure NetApp Files storage backends.

Before you begin

- Ensure that you have enabled Astra Control Provisioner on the managed Red Hat OpenShift cluster. Refer to Enable Astra Control Provisioner for instructions.
- Ensure that you have access to the `tridentctl` utility.
- Ensure that you have prepared the Azure NetApp Files storage backend for Kerberos encryption by noting the requirements and following the instructions in Azure NetApp Files documentation.
- Ensure that any NFSv4 volumes you use with Kerberos encryption are configured correctly. Refer to the NetApp NFSv4 Domain Configuration section (page 13) of the NetApp NFSv4 Enhancements and Best Practices Guide.

Create a storage backend

You can create an Azure NetApp Files storage backend configuration that includes Kerberos encryption capability.

About this task

When you create a storage backend configuration file that configures Kerberos encryption, you can define it so that it should be applied at one of two possible levels:

- The **storage backend level** using the `spec.kerberos` field
- The **virtual pool level** using the `spec.storage.kerberos` field

When you define the configuration at the virtual pool level, the pool is selected using the label in the storage class.

At either level, you can specify one of three different versions of Kerberos encryption:

- `kerberos: sec=krb5` (authentication and encryption)
- `kerberos: sec=krb5i` (authentication and encryption with identity protection)
- `kerberos: sec=krb5p` (authentication and encryption with identity and privacy protection)

Steps

1. On the managed cluster, create a storage backend configuration file using one of the following examples, depending on where you need to define the storage backend (storage backend level or virtual pool level). Replace values in brackets <> with information from your environment:

Storage backend level example

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

Virtual pool level example

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-anf-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-anf-secret
```

2. Use the configuration file you created in the previous step to create the backend:

```
tridentctl create backend -f <backend-configuration-file>
```

If the backend creation fails, something is wrong with the backend configuration. You can view the logs to determine the cause by running the following command:

```
tridentctl logs
```

After you identify and correct the problem with the configuration file, you can run the create command again.

Create a storage class

You can create a storage class to provision volumes with Kerberos encryption.

Steps

1. Create a StorageClass Kubernetes object, using the following example:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "nfs"
  selector: "type=encryption"
```

2. Create the storage class:

```
kubectl create -f sample-input/storage-class-anf-sc-nfs.yaml
```

3. Make sure that the storage class has been created:

```
kubectl get sc anf-sc-nfs
```

You should see output similar to the following:

```
NAME          PROVISIONER             AGE
anf-sc-nfs    csi.trident.netapp.io   15h
```

Provision volumes

After you create a storage backend and a storage class, you can now provision a volume. Refer to these instructions for provisioning a volume.

Recover volume data using a snapshot

Astra Control Provisioner provides rapid, in-place volume restoration from a snapshot using the `TridentActionSnapshotRestore` (TASR) CR. This CR functions as an imperative Kubernetes action and does not persist after the operation completes.

Astra Control Provisioner supports snapshot restore on the `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`, and `solidfire-san` drivers.

Before you begin

You must have a bound PVC and available volume snapshot.

- Verify the PVC status is bound.

```
kubectl get pvc
```

- Verify the volume snapshot is ready to use.

```
kubectl get vs
```

Steps

1. Create the TASR CR. This example creates a CR for PVC `pvc1` and volume snapshot `pvc1-snapshot`.

```
cat tasr-pvc1-snapshot.yaml

apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Apply the CR to restore from the snapshot. This example restores from snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

Results

Astra Control Provisioner restores the data from the snapshot. You can verify the snapshot restore status.

```
kubectl get tasr -o yaml

apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- In most cases, Astra Control Provisioner will not automatically retry the operation in case of failure. You'll need to perform the operation again.
- Kubernetes users without admin access might have to be granted permission by the admin to create a TASR CR in their application namespace.

Replicate volumes using SnapMirror

Using Astra Control Provisioner, you can create mirror relationships between a source volume on one cluster and the destination volume on the peered cluster for replicating data for disaster recovery. You can use a namespaced Custom Resource Definition (CRD) to perform the following operations:

- Create mirror relationships between volumes (PVCs)
- Remove mirror relationships between volumes
- Break the mirror relationships
- Promote the secondary volume during disaster conditions (failovers)
- Perform lossless transition of applications from cluster to cluster (during planned failovers or migrations)

Replication prerequisites

Ensure that the following prerequisites are met before you begin:

ONTAP clusters

- **Astra Control Provisioner**: Astra Control Provisioner version 23.10 or later or a supported Astra Trident must exist on both the source and destination Kubernetes clusters that utilize ONTAP as a backend.
- **Licenses**: ONTAP SnapMirror asynchronous licenses using the Data Protection bundle must be enabled on both the source and destination ONTAP clusters. Refer to SnapMirror licensing overview in ONTAP for more information.

Peering

- **Cluster and SVM**: The ONTAP storage backends must be peered. Refer to Cluster and SVM peering overview for more information.

> Ensure that the SVM names used in the replication relationship between two ONTAP clusters are unique.

- **Astra Control Provisioner and SVM**: The peered remote SVMs must be available to Astra Control Provisioner on the destination cluster.

Supported drivers

- Volume replication is supported for the ontap-nas and ontap-san drivers.

Create a mirrored PVC

Follow these steps and use the CRD examples to create mirror relationship between primary and secondary volumes.

Steps

1. Perform the following steps on the primary Kubernetes cluster:
   a. Create a StorageClass object with the `trident.netapp.io/replication: true` parameter.

      Example

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Create a PVC with previously created StorageClass.

Example

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. Create a MirrorRelationship CR with local information.

Example

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Astra Control Provisioner fetches the internal information for the volume and the volume's current data protection (DP) state, then populates the status field of the MirrorRelationship.

d. Get the TridentMirrorRelationship CR to obtain the internal name and SVM of the PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Perform the following steps on the secondary Kubernetes cluster:
   a. Create a StorageClass with the trident.netapp.io/replication: true parameter.

      Example

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        trident.netapp.io/replication: true
      ```

   b. Create a MirrorRelationship CR with destination and source information.

      Example

      ```
      kind: TridentMirrorRelationship
      apiVersion: trident.netapp.io/v1
      metadata:
        name: csi-nas
      spec:
        state: established
        volumeMappings:
        - localPVCName: csi-nas
          remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
      ```

Astra Control Provisioner will create a SnapMirror relationship with the configured relationship policy name (or default for ONTAP) and initialize it.

c. Create a PVC with previously created StorageClass to act as the secondary (SnapMirror destination).

Example

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Astra Control Provisioner will check for the TridentMirrorRelationship CRD and fail to create the volume if the relationship does not exist. If the relationship exists, Astra Control Provisioner will ensure the new FlexVol volume is placed onto an SVM that is peered with the remote SVM defined in the MirrorRelationship.

Volume Replication States

A Trident Mirror Relationship (TMR) is a CRD that represents one end of a replication relationship between PVCs. The destination TMR has a state, which tells Astra Control Provisioner what the desired state is. The destination TMR has the following states:

- **Established**: the local PVC is the destination volume of a mirror relationship, and this is a new relationship.
- **Promoted**: the local PVC is ReadWrite and mountable, with no mirror relationship currently in effect.
- **Reestablished**: the local PVC is the destination volume of a mirror relationship and was also previously in that mirror relationship.
  - The reestablished state must be used if the destination volume was ever in a relationship with the source volume because it overwrites the destination volume contents.
  - The reestablished state will fail if the volume was not previously in a relationship with the source.

Promote secondary PVC during an unplanned failover

Perform the following step on the secondary Kubernetes cluster:

- Update the *spec.state* field of TridentMirrorRelationship to `promoted`.

Promote secondary PVC during a planned failover

During a planned failover (migration), perform the following steps to promote the secondary PVC:

Steps

1. On the primary Kubernetes cluster, create a snapshot of the PVC and wait until the snapshot is created.
2. On the primary Kubernetes cluster, create the SnapshotInfo CR to obtain internal details.

   Example

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. On secondary Kubernetes cluster, update the *spec.state* field of the *TridentMirrorRelationship* CR to *promoted* and *spec.promotedSnapshotHandle* to be the internalName of the snapshot.
4. On secondary Kubernetes cluster, confirm the status (status.state field) of TridentMirrorRelationship to promoted.

Restore a mirror relationship after a failover

Before restoring a mirror relationship, choose the side that you want to make as the new primary.

Steps

1. On the secondary Kubernetes cluster, ensure that the values for the *spec.remoteVolumeHandle* field on the TridentMirrorRelationship is updated.
2. On secondary Kubernetes cluster, update the *spec.mirror* field of TridentMirrorRelationship to `reestablished`.

Additional operations

Astra Control Provisioner supports the following operations on the primary and secondary volumes:

Replicate primary PVC to a new secondary PVC

Ensure that you already have a primary PVC and a secondary PVC.

Steps

1. Delete the PersistentVolumeClaim and TridentMirrorRelationship CRDs from the established secondary (destination) cluster.
2. Delete the TridentMirrorRelationship CRD from the primary (source) cluster.
3. Create a new TridentMirrorRelationship CRD on the primary (source) cluster for the new secondary (destination) PVC you want to establish.

Resize a mirrored, primary or secondary PVC

The PVC can be resized as normal, ONTAP will automatically expand any destination flevxols if the amount of data exceeds the current size.

Remove replication from a PVC

To remove replication, perform one of the following operations on the current secondary volume:

- Delete the MirrorRelationship on the secondary PVC. This breaks the replication relationship.
- Or, update the spec.state field to *promoted*.

Delete a PVC (that was previously mirrored)

Astra Control Provisioner checks for replicated PVCs, and releases the replication relationship before attempting to delete the volume.

Delete a TMR

Deleting a TMR on one side of a mirrored relationship causes the remaining TMR to transition to *promoted* state before Astra Control Provisioner completes the deletion. If the TMR selected for deletion is already in *promoted* state, there is no existing mirror relationship and the TMR will be removed and Astra Control Provisioner will promote the local PVC to *ReadWrite*. This deletion releases SnapMirror metadata for the local volume in ONTAP. If this volume is used in a mirror relationship in the future, it must use a new TMR with an *established* volume replication state when creating the new mirror relationship.

Update mirror relationships when ONTAP is online

Mirror relationships can be updated any time after they are established. You can use the `state: promoted` or `state: reestablished` fields to update the relationships.
When promoting a destination volume to a regular ReadWrite volume, you can use *promotedSnapshotHandle* to specify a specific snapshot to restore the current volume to.

Update mirror relationships when ONTAP is offline

You can use a CRD to perform a SnapMirror update without Astra Control having direct connectivity to the ONTAP cluster. Refer to the following example format of the TridentActionMirrorUpdate:

Example

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` reflects the state of the TridentActionMirrorUpdate CRD. It can take a value from *Succeeded*, *In Progress*, or *Failed*.

Automate with Astra Control 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 Control Center UI or the Astra Control API.

To learn more, go to the Astra automation docs.

Knowledge and support

Troubleshooting

Learn how to work around some common problems you might encounter.

NetApp Knowledge Base for Astra Control

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 Discord 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 Discord 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:

- **Use the knowledge base** (login required): Search for articles, FAQs, or Break Fix information related to Astra Control.
- **Refer to the product documentation**: This is the doc site that you're currently viewing.
- **Get help via Discord**: Go to Astra in The Pub category 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 (CR) file (`astra_control_center.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 23.10 documentation
- Astra Control Center 23.07 documentation
- Astra Control Center 23.04 documentation
- Astra Control Center 22.11 documentation
- Astra Control Center 22.08 documentation
- Astra Control Center 22.04 documentation
- Astra Control Center 21.12 documentation
- Astra Control Center 21.08 documentation

Frequently asked questions

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.yaml custom resource (CR) 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.yaml CR.

Licensing

I am using an evaluation license. How do I change to the full license?

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

Steps

1. From the left navigation, select **Account** > **License**.
2. In the license overview, to the right of the license information, select the Options menu.
3. Select **Replace**.
4. Browse to the license file you downloaded and select **Add**.

I am using an evaluation license. Can I still manage apps?

Yes, you can test out the managing apps functionality with an evaluation license (including the embedded evaluation license that is installed by default). There is no difference in capabilities or features between an evaluation license and a full license; the evaluation license simply has a shorter lifespan. Refer to Licensing for more information.

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 Astra Control Provisioner (or Astra Trident) automatically uninstalled from a cluster when I unmanage it?

When you unmanage a cluster from Astra Control Center, Astra Control Provisioner or Astra Trident isn't automatically uninstalled from the cluster. To uninstall Astra Control Provisioner and its components or Astra Trident, you'll need to follow these steps to uninstall the Astra Trident instance that contains the Astra Control Provisioner service.

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. Although 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?

Astra Control Center is not shown by default as an application that you can manage, but you can back up and restore an Astra Control Center instance using another Astra Control Center instance.

Do unhealthy pods affect app management?

No, the health of pods doesn't affect app management.

Data management operations

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

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 be managed only with Astra Control.

Astra Control Provisioner

How are Astra Control Provisioner's storage provisioning features different from those in Astra Trident?

Astra Control Provisioner, as part of Astra Control, supports a superset of storage provisioning features that are unavailable in open-source Astra Trident. These features are in addition to all features that are available to the open-source Trident.

Is Astra Control Provisioner replacing Astra Trident?

Astra Control Provisioner has replaced Astra Trident as storage provisioner and orchestrator in the Astra Control architecture. Astra Control users should enable Astra Control Provisioner to use Astra Control. Astra Trident will continue to be supported in this release but will not be supported in future releases. Astra Trident will remain open source and be released, maintained, supported, and updated with new CSI and other features from NetApp. Only Astra Control Provisioner, however, that contains Astra Trident CSI functionality along with extended storage management capabilities can be used with coming Astra Control releases.

Do I have to pay for Astra Trident?

No. Astra Trident will continue to be open source and free to download. Astra Control Provisioner functionality use now requires an Astra Control license.

Can I use the storage management and provisioning features in Astra Control without installing and using all of Astra Control?

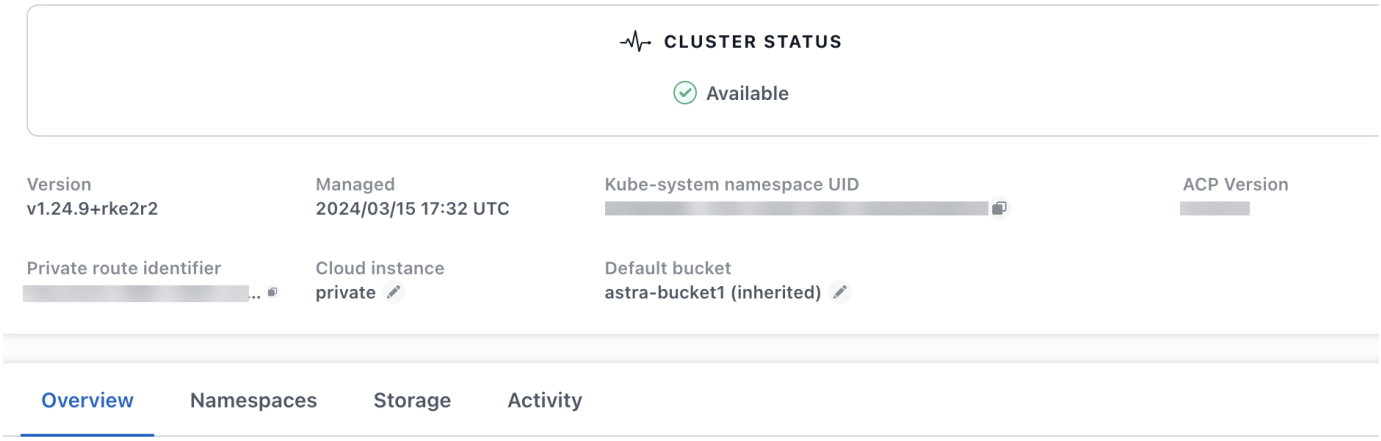
Yes, you can upgrade to Astra Control Provisioner and use its functionality even if you do not want to consume the complete feature set of Astra Control data management functionality.

How can I transition from being an existing Astra Trident user to Astra Control to use the advanced storage management and provisioning functionality?

If you are an existing Astra Trident user (this includes users of Astra Trident in the public cloud), you need to acquire an Astra Control license first. After you do, you can download the Astra Control Provisioner bundle, upgrade Astra Trident, and enable Astra Control Provisioner functionality.

How do I know if Astra Control Provisioner has replaced Astra Trident on my cluster?

After Astra Control Provisioner is installed, the host cluster in the Astra Control UI will show an `ACP version` rather than `Trident version` field and current installed version number.



If you don’t have access to the UI, you can confirm successful installation using the following methods:

Astra Trident operator

Verify the `trident-acp` container is running and that `acpVersion` is `23.10.0` or later (23.10 is the minimum version) with a status of `Installed`:

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

Response:

```
status:
  acpVersion: 24.10.0
  currentInstallationParams:
    ...
    acpImage: <my_custom_registry>/trident-acp:24.10.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl

Confirm that Astra Control Provisioner has been enabled:

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

Response:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 24.10.0 | 24.10.0 | 24.10.0. | +----------------
+----------------+-------------+
```

Legal notices

Legal notices provide access to copyright statements, trademarks, patents, and more.

Copyright

https://www.netapp.com/company/legal/copyright/

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.

https://www.netapp.com/company/legal/trademarks/

Patents

A current list of NetApp owned patents can be found at:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Privacy policy

https://www.netapp.com/company/legal/privacy-policy/

Open source

Notice files provide information about third-party copyright and licenses used in NetApp software.

- Notice for Astra Control Center

Astra Control API license

https://docs.netapp.com/us-en/astra-automation/media/astra-api-license.pdf

Copyright information

Trademark information