Get started

Astra Control Service

NetApp
March 11, 2024

Table of Contents

Get started

Learn about Astra Control

Astra Control is a Kubernetes application data lifecycle management solution that simplifies operations for stateful applications. Easily protect, back up, 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
- Clone an application 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 storage backends?	• 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-premise 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

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 Astra Trident 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 Trident 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 define apps from your cluster. Persistent volumes will be provisioned on the storage backend through 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 namespaces, 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 Trident-based storage class with an ONTAP 9.5 and above storage backend.

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

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

You can try Astra Control Center out using a 90-day evaluation license. The evaluation version is supported through email and community 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 applications to your cluster. Or, if you have some applications 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

- Documentation for the NetApp Astra product family
- Astra Control Service documentation
- Astra Control Center documentation
- Astra Trident documentation
- Use the Astra Control API
- Cloud Insights documentation
- ONTAP documentation

Supported Kubernetes deployments

Astra Control Service can manage apps that are running on a managed Kubernetes cluster in Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS). Astra Control Service can also manage clusters that you manage on your own.

- Learn how to set up Amazon Web Services for Astra Control Service.
- Learn how to set up Google Cloud for Astra Control Service.
- Learn how to set up Microsoft Azure with Azure NetApp Files for Astra Control Service.
- Learn how to set up Microsoft Azure with Azure managed disks for Astra Control Service.
- Learn how to prepare self-managed clusters before adding them to Astra Control Service.

Quick start for Astra Control Service

This page provides a high-level overview of the steps that you need to complete to get started with Astra Control Service. The links within each step take you to a page that provides more details.

[One] Set up your cloud provider

a. Google Cloud:

- Review Google Kubernetes Engine cluster requirements.
- Purchase Cloud Volumes Service for Google Cloud from the Google Cloud Marketplace.
- Enable the required APIs.
- Create a service account and service account key.
- Set up network peering from your VPC to Cloud Volumes Service for Google Cloud.

  Learn more about Google Cloud requirements.

b. Amazon Web Services:

   - Review Amazon Web Services cluster requirements.
   - Create an Amazon account.
   - Install the Amazon Web Services CLI.
   - Create an IAM user.
   - Create and attach a permissions policy.
   - Save the credentials for the IAM user.

     Learn more about Amazon Web Services requirements.

c. Microsoft Azure:

   - Review Azure Kubernetes Service cluster requirements for the storage backend you plan to use.

     Learn more about Microsoft Azure and Azure NetApp Files requirements.

     Learn more about Microsoft Azure and Azure managed disk requirements.

If you are managing your own cluster and it is not hosted by a cloud provider, review the requirements for self-managed clusters.
Learn more about self-managed cluster requirements.

[Two] Complete the Astra Control registration

a. Create a NetApp BlueXP account.
b. Specify your NetApp BlueXP email ID when creating your Astra Control account from the Astra Control product page.

Learn more about the registration process.

[Three] Add clusters to Astra Control

After you log in, select **Add cluster** to start managing your cluster with Astra Control.

Learn more about adding clusters.

Set up your cloud provider

Set up Amazon Web Services

A few steps are required to prepare your Amazon Web Services project before you can manage Amazon Elastic Kubernetes Service (EKS) clusters with Astra Control Service.

Quick start for setting up Amazon Web Services

Get started quickly by following these steps or scroll down to the remaining sections for full details.

[One] Review Astra Control Service requirements for Amazon Web Services

Ensure that clusters are healthy and running a supported version of Kubernetes, that worker nodes are online and running Linux or Windows, and more. Learn more about this step.

[Two] Create an Amazon account

If you don't already have an Amazon account, you need to create one so that you can use EKS. Learn more about this step.

[Three] Install the Amazon Web Services CLI

Install the AWS CLI so that you can manage AWS from the command line. Follow step-by-step instructions.

[Four] Optional: Create an IAM user

Create an Amazon Identity and Access Management (IAM) user. You can also skip this step and use an existing IAM user with Astra Control Service.

Read step-by-step instructions.

[Five] Create and attach a permissions policy

Create a policy with the required permissions for Astra Control Service to interact with your AWS account.

Read step-by-step instructions.

[Six] Save the credentials for the IAM user

Save the credentials for the IAM user so that you can import the credentials in to Astra Control Service.

Read step-by-step instructions.

EKS cluster requirements

A Kubernetes cluster must meet the following requirements so you can discover and manage it from Astra Control Service.

Kubernetes version

A cluster must be running a Kubernetes version in the range of 1.23 to 1.25.

Image type

The image type for each worker node must be Linux.

Cluster state

Clusters must be running in a healthy state and have at least one online worker node with no worker nodes in a failed state.

Astra Trident

Astra Trident and an external snapshot controller are required for operations with storage backends. To install them, do the following:

1. Install the snapshot CRDs and the snapshot controller.
2. Install the latest Astra Trident version.
3. Create a VolumeSnapshotClass.

CSI drivers for Amazon Elastic Block Store (EBS)

If you use the Amazon EBS storage backend, you need to install the Container Storage Interface (CSI) driver for EBS (it is not installed automatically).

Refer to the steps for instructions for installing the CSI driver.

Install an external snapshotter

If you haven't already done so, install the snapshot CRDs and the snapshot controller.

Install the CSI driver as an Amazon EKS add-on

1. Create the Amazon EBS CSI driver IAM role for service accounts. Follow the instructions in the Amazon documentation, using the AWS CLI commands in the instructions.
2. Add the Amazon EBS CSI add-on using the following AWS CLI command, replacing information in brackets <> with values specific to your environment. Replace <DRIVER_ROLE> with the name of the EBS CSI driver role that you created in the previous step:

```
aws eks create-addon \
  --cluster-name <CLUSTER_NAME> \
  --addon-name aws-ebs-csi-driver \
  --service-account-role-arn
arn:aws:iam::<ACCOUNT_ID>:role/<DRIVER_ROLE>
```

Configure the EBS storage class

1. Clone the Amazon EBS CSI driver GitHub repository to your system.

```
git clone https://github.com/kubernetes-sigs/aws-ebs-csi-
driver.git
```

2. Navigate to the dynamic-provisioning example directory.

```
cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
```

3. Deploy the ebs-sc storage class and ebs-claim persistent volume claim from the manifests directory.

```
kubectl apply -f manifests/storageclass.yaml
kubectl apply -f manifests/claim.yaml
```

4. Describe the ebs-sc storage class.

```
kubectl describe storageclass ebs-sc
```

You should see output describing the storage class attributes.

Create an Amazon account

If you don’t already have an Amazon account, you need to create one to enable billing for Amazon EKS.

Steps

1. Go to the Amazon homepage , select **Sign in** at the top right, and select **Start here**.
2. Follow the prompts to create an account.

Install the Amazon Web Services CLI

Install the AWS CLI so that you can manage AWS resources from the command line.

Step

1. Go to Getting started with the AWS CLI and follow the instructions to install the CLI.

Optional: Create an IAM user

Create an IAM user so that you can use and manage AWS services and resources with increased security. You can also skip this step, and use an existing IAM user with Astra Control Service.

Step

1. Go to Creating IAM users and follow the instructions to create an IAM user.

Create and attach a permissions policy

Create a policy with the required permissions for Astra Control Service to interact with your AWS account.

Steps

1. Create a new file called `policy.json`.
2. Copy the following JSON content into the file:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "fsx:DescribeVolumes",
                "ec2:DescribeRegions",
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject",
                "iam:SimulatePrincipalPolicy",
                "s3:ListAllMyBuckets",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:DescribeNodegroup",
                "eks:ListClusters",
                "iam:GetUser",
                "s3:DeleteObject",
                "s3:DeleteBucket",
                "autoscaling:DescribeAutoScalingGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

3. Create the policy:

```
POLICY_ARN=$(aws iam create-policy  --policy-name <policy-name> --policy
-document file://policy.json  --query='Policy.Arn' --output=text)
```

4. Attach the policy to the IAM user. Replace `<IAM-USER-NAME>` with either the user name of the IAM user you created, or an existing IAM user:

```
aws iam attach-user-policy --user-name <IAM-USER-NAME> --policy-arn
=$POLICY_ARN
```

Save the credentials for the IAM user

Save the credentials for the IAM user so that you can make Astra Control Service aware of the user.

Steps

1. Download the credentials. Replace `<IAM-USER-NAME>` with the user name of the IAM user you want to use:

   ```
   aws iam create-access-key --user-name <IAM-USER-NAME> --output json >
   credential.json
   ```

Result

The `credential.json` file is created, and you can import the credentials in to Astra Control Service.

Set up Google Cloud

A few steps are required to prepare your Google Cloud project before you can manage Google Kubernetes Engine clusters with Astra Control Service.

> If you do not start out using Google Cloud Volumes Service for Google Cloud as a storage backend but plan to use it at a later date, you should complete the necessary steps to configure Google Cloud Volumes Service for Google Cloud now. Creating a service account later means that you might lose access to your existing storage buckets.

Quick start for setting up Google Cloud

Get started quickly by following these steps or scroll down to the remaining sections for full details.

[One] Review Astra Control Service requirements for Google Kubernetes Engine

Ensure that clusters are healthy and running a supported Kubernetes version, that worker nodes are online and running a supported image type, and more. Learn more about this step.

[Two] (Optional): Purchase Cloud Volumes Service for Google Cloud

If you plan to use Cloud Volumes Service for Google Cloud as a storage backend, go to the NetApp Cloud Volumes Service page in the Google Cloud Marketplace and select Purchase. Learn more about this step.

[Three] Enable APIs in your Google Cloud project

Enable the following Google Cloud APIs:

- Google Kubernetes Engine
- Cloud Storage
- Cloud Storage JSON API
- Service Usage
- Cloud Resource Manager API
- NetApp Cloud Volumes Service

    - Required for Cloud Volumes Service for Google Cloud
    - Optional (but recommended) for Google Persistent Disk
- Service Consumer Management API
- Service Networking API
- Service Management API

Follow step-by-step instructions.

[Four] Create a service account that has the required permissions

Create a Google Cloud service account that has the following permissions:

- Kubernetes Engine Admin
- NetApp Cloud Volumes Admin
    - Required for Cloud Volumes Service for Google Cloud
    - Optional (but recommended) for Google Persistent Disk
- Storage Admin
- Service Usage Viewer
- Compute Network Viewer

Read step-by-step instructions.

[Five] Create a service account key

Create a key for the service account and save the key file in a secure location. Follow step-by-step instructions.

[Six] (Optional): Set up network peering for your VPC

If you plan to use Cloud Volumes Service for Google Cloud as a storage backend, set up network peering from your VPC to Cloud Volumes Service for Google Cloud. Follow step-by-step instructions.

GKE cluster requirements

A Kubernetes cluster must meet the following requirements so you can discover and manage it from Astra Control Service. Note that some of these requirements only apply if you plan to use Cloud Volumes Service for Google Cloud as a storage backend.

Kubernetes version

A cluster must be running a Kubernetes version in the range of 1.24 to 1.25.

Image type

The image type for each worker node must be `COS_CONTAINERD`.

Cluster state

Clusters must be running in a healthy state and have at least one online worker node with no worker nodes in a failed state.

Google Cloud region

If you plan to use Cloud Volumes Service for Google Cloud as a storage backend, clusters must be running in a Google Cloud region where Cloud Volumes Service for Google Cloud is supported. Note that Astra Control Service supports both service types: CVS and CVS-Performance. As a best practice, you should choose a region that supports Cloud Volumes Service for Google Cloud, even if you do not use it as a storage backend. This makes it easier to use Cloud Volumes Service for Google Cloud as a storage backend in the future if your performance requirements change.

Networking

If you plan to use Cloud Volumes Service for Google Cloud as a storage backend, the cluster must reside in a VPC that is peered with Cloud Volumes Service for Google Cloud. This step is described below.

Private clusters

If the cluster is private, the authorized networks must allow the Astra Control Service IP address:

52.188.218.166/32

Mode of operation for a GKE cluster

You should use the Standard mode of operation. The Autopilot mode hasn't been tested at this time. Learn more about modes of operation.

Storage pools

If you use NetApp Cloud Volumes Service as a storage backend with the CVS service type, you need to configure storage pools before you can provision volumes. Refer to Service type, storage classes, and PV size for GKE clusters for more information.

Optional: Purchase Cloud Volumes Service for Google Cloud

Astra Control Service can use Cloud Volumes Service for Google Cloud as the storage backend for your persistent volumes. If you plan to use this service, you need to purchase Cloud Volumes Service for Google Cloud from the Google Cloud Marketplace to enable billing for persistent volumes.

Step

1. Go to the NetApp Cloud Volumes Service page in the Google Cloud Marketplace, select **Purchase**, and follow the prompts.

   Follow step-by-step instructions in the Google Cloud documentation to purchase and enable the service.

Enable APIs in your project

Your project needs permissions to access specific Google Cloud APIs. APIs are used to interact with Google Cloud resources, such as Google Kubernetes Engine (GKE) clusters and NetApp Cloud Volumes Service storage.

Step

1. Use the Google Cloud console or gcloud CLI to enable the following APIs:

   - Google Kubernetes Engine
   - Cloud Storage
   - Cloud Storage JSON API
   - Service Usage

- Cloud Resource Manager API
- NetApp Cloud Volumes Service (Required for Cloud Volumes Service for Google Cloud)
- Service Consumer Management API
- Service Networking API
- Service Management API

The following video shows how to enable the APIs from the Google Cloud console.

► https://docs.netapp.com/us-en/astra-control-service/media/get-started/video-enable-gcp-apis.mp4 *(video)*

Create a service account

Astra Control Service uses a Google Cloud service account to facilitate Kubernetes application data management on your behalf.

Steps

1. Go to Google Cloud and create a service account by using the console, gcloud command, or another preferred method.
2. Grant the service account the following roles:
   - **Kubernetes Engine Admin** - Used to list clusters and create admin access to manage apps.
   - **NetApp Cloud Volumes Admin** - Used to manage persistent storage for apps.
   - **Storage Admin** - Used to manage buckets and objects for backups of apps.
   - **Service Usage Viewer** - Used to check if the required Cloud Volumes Service for Google Cloud APIs are enabled.
   - **Compute Network Viewer** - Used to check if the Kubernetes VPC is allowed to reach Cloud Volumes Service for Google Cloud.

If you’d like to use gcloud, you can follow steps from within the Astra Control interface. Select **Account > Credentials > Add Credentials**, and then select **Instructions**.

If you’d like to use the Google Cloud console, the following video shows how to create the service account from the console.

► https://docs.netapp.com/us-en/astra-control-service/media/get-started/video-create-gcp-service-account.mp4 *(video)*

Configure the service account for a shared VPC

To manage GKE clusters that reside in one project, but use a VPC from a different project (a shared VPC), then you need to specify the Astra service account as a member of the host project with the **Compute Network Viewer** role.

Steps

1. From the Google Cloud console, go to **IAM & Admin** and select **Service Accounts**.
2. Find the Astra service account that has the required permissions and then copy the email address.
3. Go to your host project and then select **IAM & Admin** > **IAM**.
4. Select **Add** and add an entry for the service account.
   a. **New members**: Enter the email address for the service account.
   b. **Role**: Select **Compute Network Viewer**.
   c. Select **Save**.

Result

Adding a GKE cluster using a shared VPC will fully work with Astra.

Create a service account key

Instead of providing a user name and password to Astra Control Service, you'll provide a service account key when you add your first cluster. Astra Control Service uses the service account key to establish the identity of the service account that you just set up.

The service account key is plaintext stored in the JavaScript Object Notation (JSON) format. It contains information about the GCP resources that you have permission to access.

You can only view or download the JSON file when you create the key. However, you can create a new key at any time.

Steps

1. Go to Google Cloud and create a service account key by using the console, gcloud command, or another preferred method.
2. When prompted, save the service account key file in a secure location.

The following video shows how to create the service account key from the Google Cloud console.

► https://docs.netapp.com/us-en/astra-control-service/media/get-started/video-create-gcp-service-account-key.mp4 *(video)*

Optional: Set up network peering for your VPC

If you plan to use Cloud Volumes Service for Google Cloud as a storage backend service, the final step is to set up networking peering from your VPC to Cloud Volumes Service for Google Cloud.

The easiest way to set up network peering is by obtaining the gcloud commands directly from Cloud Volumes Service. The commands are available from Cloud Volumes Service when creating a new file system.

Steps

1. Go to NetApp BlueXP Global Regions Maps and identify the service type that you'll be using in the Google Cloud region where your cluster resides.

   Cloud Volumes Service provides two service types: CVS and CVS-Performance. Learn more about these service types.

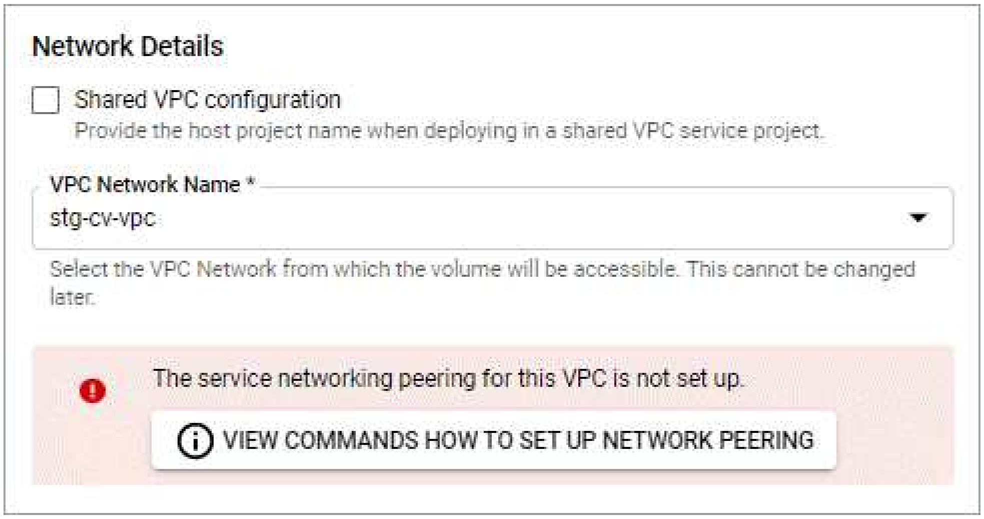
2. Go to Cloud Volumes in Google Cloud Platform.
3. On the **Volumes** page, select **Create**.
4. Under **Service Type**, select either **CVS** or **CVS-Performance**.

   You need to choose the correct service type for your Google Cloud region. This is the service type that you identified in step 1. After you select a service type, the list of regions on the page updates with the regions where that service type is supported.

   After this step, you'll only need to enter your networking information to obtain the commands.

5. Under **Region**, select your region and zone.
6. Under **Network Details**, select your VPC.

   If you haven't set up network peering, you'll see the following notification:



Network Details

□ Shared VPC configuration
Provide the host project name when deploying in a shared VPC service project.

VPC Network Name *
stg-cv-vpc
Select the VPC Network from which the volume will be accessible. This cannot be changed later.

The service networking peering for this VPC is not set up.

7. Select the button to view the network peering set up commands.
8. Copy the commands and run them in Cloud Shell.

   For more details about using these commands, refer to the Quickstart for Cloud Volumes Service for GCP.

Learn more about configuring private services access and setting up network peering.

9. After you're done, you can select cancel on the **Create File System** page.

   We started creating this volume only to get the commands for network peering.

Set up Microsoft Azure with Azure NetApp Files

A few steps are required to prepare your Microsoft Azure subscription before you can manage Azure Kubernetes Service clusters with Astra Control Service. Follow these instructions if you plan to use Azure NetApp Files as a storage backend.

Quick start for setting up Azure

Get started quickly by following these steps or scroll down to the remaining sections for full details.

[One] Review Astra Control Service requirements for Azure Kubernetes Service

Ensure that clusters are healthy and running a supported version of Kubernetes, that node pools are online and running Linux, and more. Learn more about this step.

[Two] Sign up for Microsoft Azure

Create a Microsoft Azure account. Learn more about this step.

[Three] Register for Azure NetApp Files

Register the NetApp Resource Provider. Learn more about this step.

[Four] Create a NetApp account

Go to Azure NetApp Files in the Azure portal and create a NetApp account. Learn more about this step.

[Five] Set up capacity pools

Set up one or more capacity pools for your persistent volumes. Learn more about this step.

[Six] Delegate a subnet to Azure NetApp Files

Delegate a subnet to Azure NetApp Files so that Astra Control Service can create persistent volumes in that subnet. Learn more about this step.

[Seven] Create an Azure service principal

Create an Azure service principal that has the Contributor role. Learn more about this step.

[Eight] Optional: Configure redundancy for Azure backup buckets

By default, the buckets Astra Control Service uses to store Azure Kubernetes Service backups use the Locally Redundant Storage (LRS) redundancy option. As an optional step, you can configure a more durable level of redundancy for Azure buckets. Learn more about this step.

Azure Kubernetes Service cluster requirements

A Kubernetes cluster must meet the following requirements so you can discover and manage it from Astra Control Service.

Kubernetes version

Clusters must be running Kubernetes version 1.23 to 1.25.

Image type

The image type for all node pools must be Linux.

Cluster state

Clusters must be running in a healthy state and have at least one online worker node with no worker nodes in a failed state.

Azure region

Clusters must reside in a region where Azure NetApp Files is available. View Azure products by region.

Subscription

Clusters must reside in a subscription where Azure NetApp Files is enabled. You'll choose a subscription when you register for Azure NetApp Files.

VNet

Consider the following VNet requirements:

- Clusters must reside in a VNet that has direct access to an Azure NetApp Files delegated subnet. Learn how to set up a delegated subnet.
- If your Kubernetes clusters are in a VNet that's peered to the Azure NetApp Files delegated subnet that's in another VNet, then both sides of the peering connection must be online.
- Be aware that the default limit for the number of IPs used in a VNet (including immediately peered VNets) with Azure NetApp Files is 1,000. View Azure NetApp Files resource limits.

  If you're close to the limit, you have two options:

  - You can submit a request for a limit increase. Contact your NetApp representative if you need help.
  - When creating a new Amazon Kubernetes Service (AKS) cluster, specify a new network for the cluster. Once the new network is created, provision a new subnet and delegate the subnet to Azure NetApp Files.

Sign up for Microsoft Azure

If you don't have a Microsoft Azure account, begin by signing up for Microsoft Azure.

Steps

1. Go to the Azure subscription page to subscribe to the Azure service.
2. Select a plan and follow the instructions to complete the subscription.

Register for Azure NetApp Files

Get access to Azure NetApp Files by registering the NetApp Resource Provider.

Steps

1. Log in to the Azure portal.
2. Follow Azure NetApp Files documentation to register the NetApp Resource Provider.

Create a NetApp account

Create a NetApp account in Azure NetApp Files.

Step

1. Follow Azure NetApp Files documentation to create a NetApp account from the Azure portal.

Set up a capacity pool

One or more capacity pools are required so that Astra Control Service can provision persistent volumes in a capacity pool. Astra Control Service doesn’t create capacity pools for you.

Take the following into consideration as you set up capacity pools for your Kubernetes apps:

- The capacity pools need to be created in the same Azure region where the AKS clusters will be managed with Astra Control Service.
- A capacity pool can have an Ultra, Premium, or Standard service level. Each of these service levels are designed for different performance needs. Astra Control Service supports all three.

  You need to set up a capacity pool for each service level that you want to use with your Kubernetes clusters.

  Learn more about service levels for Azure NetApp Files.

- Before you create a capacity pool for the apps that you intend to protect with Astra Control Service, choose the required performance and capacity for those apps.

  Provisioning the right amount of capacity ensures that users can create persistent volumes as they are needed. If capacity isn’t available, then the persistent volumes can’t be provisioned.

- An Azure NetApp Files capacity pool can use the manual or auto QoS type. Astra Control Service supports auto QoS capacity pools. Manual QoS capacity pools aren’t supported.

Step

1. Follow Azure NetApp Files documentation to set up an auto QoS capacity pool.

Delegate a subnet to Azure NetApp Files

You need to delegate a subnet to Azure NetApp Files so that Astra Control Service can create persistent volumes in that subnet. Note that Azure NetApp Files enables you to have only one delegated subnet in a VNet.

If you’re using peered VNets, then both sides of the peering connection must be online: the VNet where your Kubernetes clusters reside and the VNet that has the Azure NetApp Files delegated subnet.

Step

1. Follow the Azure NetApp Files documentation to delegate a subnet to Azure NetApp Files.

After you’re done

Wait about 10 minutes before discovering the cluster running in the delegated subnet.

Create an Azure service principal

Astra Control Service requires a Azure service principal that is assigned the Contributor role. Astra Control Service uses this service principal to facilitate Kubernetes application data management on your behalf.

A service principal is an identity created specifically for use with applications, services, and tools. Assigning a role to the service principal restricts access to specific Azure resources.

Follow the steps below to create a service principal using the Azure CLI. You'll need to save the output in a JSON file and provide it to Astra Control Service later on. Refer to Azure documentation for more details about using the CLI.

The following steps assume that you have permission to create a service principal and that you have the Microsoft Azure SDK (az command) installed on your machine.

Requirements

- The service principal must use regular authentication. Certificates aren't supported.
- The service principal must be granted Contributor or Owner access to your Azure subscription.
- The subscription or resource group you choose for scope must contain the AKS clusters and your Azure NetApp Files account.

Steps

1. Identify the subscription and tenant ID where your AKS clusters reside (these are the clusters that you want to manage in Astra Control Service).

   ```
   az configure --list-defaults
   az account list --output table
   ```

2. Do one of the following, depending on if you use an entire subscription or a resource group:

   - Create the service principal, assign the Contributor role, and specify the scope to the entire subscription where the clusters reside.

     ```
     az ad sp create-for-rbac --name service-principal-name --role
     contributor --scopes /subscriptions/SUBSCRIPTION-ID
     ```

   - Create the service principal, assign the Contributor role, and specify the resource group where the clusters reside.

     ```
     az ad sp create-for-rbac --name service-principal-name --role
     contributor --scopes /subscriptions/SUBSCRIPTION-
     ID/resourceGroups/RESOURCE-GROUP-ID
     ```

3. Store the resulting Azure CLI output as a JSON file.

   You'll need to provide this file so that Astra Control Service can discover your AKS clusters and manage Kubernetes data management operations. Learn about managing credentials in Astra Control Service.

4. Optional: Add the subscription ID to the JSON file so that Astra Control Service automatically populates the ID when you select the file.

   Otherwise, you’ll need to enter the subscription ID in Astra Control Service when prompted.

   Example

```
{
  "appId": "0db3929a-bfb0-4c93-baee-aaf8",
  "displayName": "sp-example-dev-sandbox",
  "name": "http://sp-example-dev-sandbox",
  "password": "mypassword",
  "tenant": "011cdf6c-7512-4805-aaf8-7721afd8ca37",
  "subscriptionId": "99ce999a-8c99-99d9-a9d9-99cce99f99ad"
}
```

5. Optional: Test your service principal. Choose from the following example commands depending on the scope your service principal uses.

   Subscription scope

```
az login --service-principal --username APP-ID-SERVICEPRINCIPAL
--password PASSWORD --tenant TENANT-ID
az group list --subscription SUBSCRIPTION-ID
az aks list --subscription SUBSCRIPTION-ID
az storage container list --account-name STORAGE-ACCOUNT-NAME
```

   Resource group scope

```
az login --service-principal --username APP-ID-SERVICEPRINCIPAL
--password PASSWORD --tenant TENANT-ID
az aks list --subscription SUBSCRIPTION-ID --resource-group RESOURCE-
GROUP-ID
```

Optional: Configure redundancy for Azure backup buckets

You can configure a more durable redundancy level for Azure backup buckets. By default, the buckets Astra Control Service uses to store Azure Kubernetes Service backups use the Locally Redundant Storage (LRS) redundancy option. To use a more durable redundancy option for Azure buckets, you need to do the following:

Steps

1. Create an Azure storage account that uses the redundancy level you need using these instructions.
2. Create an Azure container in the new storage account using these instructions.
3. Add the container as a bucket to Astra Control Service. Refer to Add an additional bucket.
4. (Optional) To use the newly created bucket as the default bucket for Azure backups, set it as the default bucket for Azure. Refer to Change the default bucket.

Set up Microsoft Azure with Azure managed disks

A few steps are required to prepare your Microsoft Azure subscription before you can manage Azure Kubernetes Service clusters with Astra Control Service. Follow these instructions if you plan to use Azure managed disks as a storage backend.

Quick start for setting up Azure

Get started quickly by following these steps or scroll down to the remaining sections for full details.

[One] Review Astra Control Service requirements for Azure Kubernetes Service

Ensure that clusters are healthy and running a supported version of Kubernetes, that node pools are online and running Linux, and more. Learn more about this step.

[Two] Sign up for Microsoft Azure

Create a Microsoft Azure account. Learn more about this step.

[Three] Create an Azure service principal

Create an Azure service principal that has the Contributor role. Learn more about this step.

[Four] Configure Container Storage Interface (CSI) driver details

You need to configure your Azure subscription and the cluster to work with the CSI drivers. Learn more about this step.

[Five] Optional: Configure redundancy for Azure backup buckets

By default, the buckets Astra Control Service uses to store Azure Kubernetes Service backups use the Locally Redundant Storage (LRS) redundancy option. As an optional step, you can configure a more durable level of redundancy for Azure buckets. Learn more about this step.

Azure Kubernetes Service cluster requirements

A Kubernetes cluster must meet the following requirements so you can discover and manage it from Astra Control Service.

Kubernetes version

Clusters must be running Kubernetes version 1.24 to 1.26.

Image type

The image type for all node pools must be Linux.

Cluster state

Clusters must be running in a healthy state and have at least one online worker node with no worker nodes in a failed state.

Azure region

As a best practice, you should choose a region that supports Azure NetApp Files, even if you do not use it as a storage backend. This makes it easier to use Azure NetApp Files as a storage backend in the future if your performance requirements change. View Azure products by region.

CSI drivers

Clusters must have the appropriate CSI drivers installed.

Sign up for Microsoft Azure

If you don’t have a Microsoft Azure account, begin by signing up for Microsoft Azure.

Steps

1. Go to the Azure subscription page to subscribe to the Azure service.
2. Select a plan and follow the instructions to complete the subscription.

Create an Azure service principal

Astra Control Service requires a Azure service principal that is assigned the Contributor role. Astra Control Service uses this service principal to facilitate Kubernetes application data management on your behalf.

A service principal is an identity created specifically for use with applications, services, and tools. Assigning a role to the service principal restricts access to specific Azure resources.

Follow the steps below to create a service principal using the Azure CLI. You’ll need to save the output in a JSON file and provide it to Astra Control Service later on. Refer to Azure documentation for more details about using the CLI.

The following steps assume that you have permission to create a service principal and that you have the Microsoft Azure SDK (az command) installed on your machine.

Requirements

- The service principal must use regular authentication. Certificates aren’t supported.
- The service principal must be granted Contributor or Owner access to your Azure subscription.
- The subscription or resource group you choose for scope must contain the AKS clusters and your Azure NetApp Files account.

Steps

1. Identify the subscription and tenant ID where your AKS clusters reside (these are the clusters that you want to manage in Astra Control Service).

   ```
   az configure --list-defaults
   az account list --output table
   ```

2. Do one of the following, depending on if you use an entire subscription or a resource group:
   - Create the service principal, assign the Contributor role, and specify the scope to the entire subscription where the clusters reside.

     ```
     az ad sp create-for-rbac --name service-principal-name --role
     contributor --scopes /subscriptions/SUBSCRIPTION-ID
     ```

   - Create the service principal, assign the Contributor role, and specify the resource group where the clusters reside.

```
az ad sp create-for-rbac --name service-principal-name --role
contributor --scopes /subscriptions/SUBSCRIPTION-
ID/resourceGroups/RESOURCE-GROUP-ID
```

3. Store the resulting Azure CLI output as a JSON file.

   You’ll need to provide this file so that Astra Control Service can discover your AKS clusters and manage Kubernetes data management operations. Learn about managing credentials in Astra Control Service.

4. Optional: Add the subscription ID to the JSON file so that Astra Control Service automatically populates the ID when you select the file.

   Otherwise, you’ll need to enter the subscription ID in Astra Control Service when prompted.

   Example

   ```
   {
     "appId": "0db3929a-bfb0-4c93-baee-aaf8",
     "displayName": "sp-example-dev-sandbox",
     "name": "http://sp-example-dev-sandbox",
     "password": "mypassword",
     "tenant": "011cdf6c-7512-4805-aaf8-7721afd8ca37",
     "subscriptionId": "99ce999a-8c99-99d9-a9d9-99cce99f99ad"
   }
   ```

5. Optional: Test your service principal. Choose from the following example commands depending on the scope your service principal uses.

   Subscription scope

   ```
   az login --service-principal --username APP-ID-SERVICEPRINCIPAL
   --password PASSWORD --tenant TENANT-ID
   az group list --subscription SUBSCRIPTION-ID
   az aks list --subscription SUBSCRIPTION-ID
   az storage container list --account-name STORAGE-ACCOUNT-NAME
   ```

   Resource group scope

   ```
   az login --service-principal --username APP-ID-SERVICEPRINCIPAL
   --password PASSWORD --tenant TENANT-ID
   az aks list --subscription SUBSCRIPTION-ID --resource-group RESOURCE-
   GROUP-ID
   ```

Configure Container Storage Interface (CSI) driver details

To use Azure managed disks with Astra Control Service, you’ll need to install the required CSI drivers.

Enable the CSI driver feature in your Azure subscription

Before you install the CSI drivers, you need to enable the CSI driver feature in your Azure subscription.

Steps

1. Open the Azure command line interface.
2. Run the following command to register the driver:

   ```
   az feature register --namespace "Microsoft.ContainerService" --name
   "EnableAzureDiskFileCSIDriver"
   ```

3. Run the following command to ensure the change is propagated:

   ```
   az provider register -n Microsoft.ContainerService
   ```

   You should see output similar to the following:

```
{
"id": "/subscriptions/b200155f-001a-43be-87be-
3edde83acef4/providers/Microsoft.Features/providers/Microsoft.ContainerSer
vice/features/EnableAzureDiskFileCSIDriver",
"name": "Microsoft.ContainerService/EnableAzureDiskFileCSIDriver",
"properties": {
   "state": "Registering"
},
"type": "Microsoft.Features/providers/features"
}
```

Install the Azure managed disk CSI drivers in your Azure Kubernetes Service cluster

You can install the Azure CSI drivers to complete your preparation.

Step

1. Go to the Microsoft CSI driver documentation.
2. Follow the instructions to install the required CSI drivers.

Optional: Configure redundancy for Azure backup buckets

You can configure a more durable redundancy level for Azure backup buckets. By default, the buckets Astra Control Service uses to store Azure Kubernetes Service backups use the Locally Redundant Storage (LRS) redundancy option. To use a more durable redundancy option for Azure buckets, you need to do the following:

Steps

1. Create an Azure storage account that uses the redundancy level you need using these instructions.
2. Create an Azure container in the new storage account using these instructions.
3. Add the container as a bucket to Astra Control Service. Refer to Add an additional bucket.
4. (Optional) To use the newly created bucket as the default bucket for Azure backups, set it as the default bucket for Azure. Refer to Change the default bucket.

Register for an Astra Control Service account

To use Astra Control Service, you need an Astra Control Service account that is associated with your NetApp BlueXP account. Complete the Astra Control Service registration process and then, if you don't already have a BlueXP account, sign up to BlueXP to access Astra Control Service.

Register for an Astra Control account

Before you can log in to Astra Control Service, you need to complete a registration process to obtain an Astra Control Service account.

When you use Astra Control Service, you'll manage your apps from within an account. An account includes users who can view and manage the apps within the account, as well as your billing details.

Steps

1. Go to the Astra Control page on BlueXP.
2. Select **Get started now**.
3. Provide the required information in the form.

   A few important things to note as you fill out the form:

   - Your business name and address must be accurate because we verify them to meet the requirements of Global Trade Compliance.
   - The **Astra Account Name** is the name of your business's Astra Control Service account. You'll see this name in the Astra Control Service user interface. Note that you can create additional accounts (up to 5), if needed.
   - In the **Business Email Address** field, if you have a NetApp BlueXP account, enter the email you use for that account here. If you don't yet have a NetApp BlueXP account, use the email address you enter here when you sign up to BlueXP.
4. Select **Create Account**.

Sign up to BlueXP

Astra Control Service is integrated within NetApp BlueXP's authentication service. You can log in to NetApp BlueXP using your BlueXP or NetApp Support Site credentials. If you don't already have a NetApp BlueXP or NetApp Support Site account, sign up to BlueXP so you can access Astra Control Service and NetApp's other cloud services. If you already have a BlueXP or NetApp Support Site account and have completed registration, you can access Astra Control Service directly using your BlueXP or NetApp Support Site credentials.

> You can also use single sign-on to log in to BlueXP using credentials from your corporate directory (federated identity). To learn more, go to the Help Center and then select **Cloud Central sign-in options**.

Steps

1. Go to NetApp BlueXP.
2. In the top right, select **Get Started**.
3. Select **Sign up**.
4. Fill out the form.

   Ensure that the phone number and email address you enter here are the same that you used in the preceding Astra Control registration form.

5. Select **Sign up**.

   > The email address that you enter in these forms is for your NetApp BlueXP user ID. Use this BlueXP user ID when you sign up for a new Astra Control account, or when an Astra Control admin invites you to an existing Astra Control account.

6. Wait for an email from NetApp BlueXP. The email comes from the address saas.support@netapp.com, and might take several minutes to arrive. Be sure to check your spam folder.
7. When the email arrives, select the link in the email to verify your email address.

Result

You now have an active BlueXP user login.

Now that you're registered, you can access Astra Control directly using your BlueXP credentials from https://astra.netapp.io.

Add a cluster to Astra Control Service

After you set up your environment, you're ready to create a Kubernetes cluster and then add it to Astra Control Service. This enables you to use Astra Control Service to protect your applications on the cluster.

Depending on the type of cluster you need to add to Astra Control Service, you need to use different steps to add the cluster.

- Add a public provider-managed cluster to Astra Control Service: Use these steps to add a cluster that has a public IP address and is managed by a cloud provider. You will need the Service Principal account, service account, or user account for the cloud provider.
- Add a private provider-managed cluster to Astra Control Service: Use these steps to add a cluster that has a private IP address and is managed by a cloud provider. You will need the Service Principal account, service account, or user account for the cloud provider.
- Add a public self-managed cluster to Astra Control Service: Use these steps to add a cluster that has a public IP address and is managed by your organization. You will need to create a kubeconfig file for the cluster you want to add.
- Add a private self-managed cluster to Astra Control Service: Use these steps to add a cluster that has a private IP address and is managed by your organization. You will need to create a kubeconfig file for the

cluster you want to add.

Install Astra Connector for private clusters

Astra Control Service uses Astra Connector to enable communication between Astra Control Service and private clusters. You need to install Astra Connector on private clusters that you want to manage.

Astra Connector supports the following types of private clusters:

- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)
- Red Hat OpenShift Service on AWS (ROSA)
- ROSA with AWS PrivateLink
- Red Hat OpenShift Container Platform on-premise

Install Astra Connector

About this task

- When you perform these steps, execute these commands against the private cluster that you want to manage with Astra Control Service.
- 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 private cluster you want to manage with Astra Control Service.
- You need Kubernetes administrator permissions to install the Astra Connector operator on the cluster.

Steps

1. Install the Astra Connector operator on the private cluster you want to manage with Astra Control Service. 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/23.07.0-
   202310251519/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 the astra-connector namespace:

```
kubectl create ns astra-connector
```

5. 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_SERVICE_URL>**: The web UI URL of Astra Control Service. For example:

     ```
     https://astra.netapp.io
     ```

   - **<ASTRA_CONTROL_SERVICE_API_TOKEN>**: The Astra Control API token you obtained in the preceding step.
   - **<PRIVATE_AKS_CLUSTER_NAME>**: (AKS clusters only) - The cluster name of the private Azure Kubernetes Service cluster. Uncomment and populate this line only if you are adding a private AKS cluster.
   - **<ASTRA_CONTROL_ACCOUNT_ID>**: Obtained from the Astra Control web UI. Select the figure icon at the top right of the page and select **API access**.

     ```
     apiVersion: netapp.astraconnector.com/v1
     kind: AstraConnector
     metadata:
       name: astra-connector
       namespace: astra-connector
     spec:
       natssync-client:
         cloud-bridge-url: <ASTRA_CONTROL_SERVICE_URL>
       imageRegistry:
         name: theotw
         secret: ""
       astra:
         token: <ASTRA_CONTROL_SERVICE_API_TOKEN>
         #clusterName: <PRIVATE_AKS_CLUSTER_NAME>
         accountId: <ASTRA_CONTROL_ACCOUNT_ID>
         acceptEULA: yes
     ```

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

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

7. Verify that the Astra Connector is fully deployed:

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

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

```
kubectl get astraconnector -n astra-connector
```

You should see output similar to the following:

```
NAME                 REGISTERED   ASTRACONNECTORID
STATUS
astra-connector      true         be475ae5-1511-4eaa-9b9e-712f09b0d065
Registered with Astra
```

Make note of the ASTRACONNECTORID; you will need it when you add the cluster to Astra Control.

What's next?

Now that you've installed Astra Connector, you're ready to add your private cluster to Astra Control Service.

- Add a private provider-managed cluster to Astra Control Service: Use these steps to add a cluster that has a private IP address and is managed by a cloud provider. You will need the Service Principal account, service account, or user account for the cloud provider.
- Add a private self-managed cluster to Astra Control Service: Use these steps to add a cluster that has a private IP address and is managed by your organization. You will need to create a kubeconfig file for the cluster you want to add.

For more information

- Add a cluster

Add a provider-managed cluster

Add a public provider-managed cluster to Astra Control Service

After you set up your cloud environment, you're ready to create a Kubernetes cluster and then add it to Astra Control Service.

- Create a Kubernetes cluster
- Add the cluster to Astra Control Service
- Change the default storage class

Create a Kubernetes cluster

If you don't have a cluster yet, you can create one that meets the requirements of one of the following providers:

- Astra Control Service requirements for Azure Kubernetes Service (AKS) with Azure NetApp Files
- Astra Control Service requirements for Azure Kubernetes Service (AKS) with Azure managed disks

- Astra Control Service requirements for Google Kubernetes Engine (GKE)
- Astra Control Service requirements for Amazon Elastic Kubernetes Service (EKS)

> ⓘ Astra Control Service supports AKS clusters that use Azure Active Directory (Azure AD) for authentication and identity management. When you create the cluster, follow the instructions in the official documentation to configure the cluster to use Azure AD. You'll need to make sure your clusters meet the requirements for AKS-managed Azure AD integration.

Add the cluster to Astra Control Service

After you log in to Astra Control Service, your first step is to start managing your clusters. Before you add a cluster to Astra Control Service, you'll need to perform specific tasks and make sure the cluster meets certain requirements.

When you manage Azure Kubernetes Service and Google Kubernetes Engine clusters, note that you have two options for Astra Trident installation and lifecycle management:

- You can use Astra Control Service to automatically manage the lifecycle of Astra Trident. To do this, make sure that Astra Trident is not installed on the cluster that you want to manage with Astra Control Service. In this case, Astra Trident automatically installs Astra Trident when you begin managing the cluster, and Astra Trident upgrades are handled automatically.
- You can manage the lifecycle of Astra Trident yourself. To do this, install Astra Trident on the cluster before managing the cluster with Astra Control Service. In this case, Astra Control Service detects that Astra Trident is already installed and does not reinstall it or manage Astra Trident upgrades. Refer to the Astra Trident documentation for installation instructions.

When you manage Amazon Web Services clusters with Astra Control Service, if you need storage backends that are enabled by Astra Trident, you need to install Astra Trident manually on the cluster before you manage it with Astra Control Service. Refer to the Astra Trident documentation for installation instructions.

Before you begin

Amazon Web Services

- You should have the JSON file containing the credentials of the IAM user that created the cluster. Learn how to create an IAM user.
- Astra Trident is required for Amazon FSx for NetApp ONTAP. If you plan to use Amazon FSx for NetApp ONTAP as a storage backend for your EKS cluster, refer to the Astra Trident information in the EKS cluster requirements.
- (Optional) If you need to provide provide `kubectl` command access for a cluster to other IAM users that are not the cluster's creator, refer to the instructions in How do I provide access to other IAM users and roles after cluster creation in Amazon EKS?.
- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Amazon Web Services. Refer to the Cloud Volumes ONTAP setup documentation.

Microsoft Azure

- You should have the JSON file that contains the output from the Azure CLI when you created the service principal. Learn how to set up a service principal.

  You'll also need your Azure subscription ID, if you didn't add it to the JSON file.
- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Microsoft Azure. Refer to the Cloud Volumes ONTAP setup documentation.

Google Cloud

- You should have the service account key file for a service account that has the required permissions. Learn how to set up a service account.
- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Google Cloud. Refer to the Cloud Volumes ONTAP setup documentation.

Steps

1. (Optional) If you are adding an Amazon EKS cluster or want to manage the installation and upgrades of Astra Trident yourself, install Astra Trident on the cluster. Refer to the Astra Trident documentation for installation instructions.
2. Open the Astra Control Service web UI in a browser.
3. On the Dashboard, select **Manage Kubernetes cluster**.

   Follow the prompts to add the cluster.
4. **Provider**: Select your cloud provider and then either provide the required credentials to create a new cloud instance, or select an existing cloud instance to use.
   a. **Amazon Web Services**: Provide details about your Amazon Web Services IAM user account by uploading a JSON file or by pasting the contents of that JSON file from your clipboard.

      The JSON file should contain the credentials of the IAM user that created the cluster.

   b. **Microsoft Azure**: Provide details about your Azure service principal by uploading a JSON file or by pasting the contents of that JSON file from your clipboard.

The JSON file should contain the output from the Azure CLI when you created the service principal. It can also include your subscription ID so it's automatically added to Astra. Otherwise, you need to manually enter the ID after providing the JSON.

c. **Google Cloud Platform**: Provide the service account key file either by uploading the file or by pasting the contents from your clipboard.

Astra Control Service uses the service account to discover clusters running in Google Kubernetes Engine.

d. **Other**: This tab is for use with self-managed clusters only.

5. **Cloud instance name**: Provide a name for the new cloud instance that will be created when you add this cluster. Learn more about cloud instances.
6. Select **Next**.

   Astra Control Service displays a list of clusters that you can choose from.

7. **Cluster**: Select a cluster from the list to add to Astra Control Service.

   > ⓘ When you are selecting from the list of clusters, pay careful attention to the **Eligiblity** column. If a cluster is "Ineligible" or "Partially eligible", hover over the status to determine if there's an issue with the cluster. For example, it might identify that the cluster doesn't have a worker node.

8. Select **Next**.
9. (Optional) **Storage**: Optionally, select the storage class that you'd like Kubernetes applications deployed to this cluster to use by default.

   a. To select a new default storage class for the cluster, enable the **Assign a new default storage class** check box.

   b. Select a new default storage class from the list.

   > ⓘ Each cloud provider storage service displays the following price, performance, and resilience information:
   >
   > - Cloud Volumes Service for Google Cloud: Price, performance, and resilience information
   > - Google Persistent Disk: No price, performance, or resilience information available
   > - Azure NetApp Files: Performance and resilience information
   > - Azure Managed disks: No price, performance, or resilience information available
   > - Amazon Elastic Block Store: No price, performance, or resilience information available
   > - Amazon FSx for NetApp ONTAP: No price, performance, or resilience information available
   > - NetApp Cloud Volumes ONTAP: No price, performance, or resilience information available

   Each storage class can utilize one of the following services:

   - Cloud Volumes Service for Google Cloud

- Google Persistent Disk
- Azure NetApp Files
- Azure managed disks
- Amazon Elastic Block Store
- Amazon FSx for NetApp ONTAP
- NetApp Cloud Volumes ONTAP

  Learn more about storage classes for Amazon Web Services clusters, storage classes for GKE clusters, and storage classes for AKS clusters.

10. Select **Next**.
11. **Review & Approve**: Review the configuration details.
12. Select **Add** to add the cluster to Astra Control Service.

Result

If this is the first cluster that you have added for this cloud provider, Astra Control Service creates an object store for the cloud provider for backups of applications running on eligible clusters. (When you add subsequent clusters for this cloud provider, no further object stores are created.) If you specified a default storage class, Astra Control Service sets the default storage class that you specified. For clusters managed in Amazon Web Services or Google Cloud Platform, Astra Control Service also creates an admin account on the cluster. These actions can take several minutes.

Change the default storage class

You can change the default storage class for a cluster.

Change the default storage class using Astra Control

You can change the default storage class for a cluster from within Astra Control. If your cluster uses a previously installed storage backend service, you might not be able to use this method to change the default storage class (the **Set as default** action is not selectable). In this case, you can Change the default storage class using the command line.

Steps

1. In the Astra Control Service 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**.

Change the default storage class using the command line

You can change the default storage class for a cluster using Kubernetes commands. This method works regardless of your cluster’s configuration.

Steps

1. Log in to your Kubernetes cluster.

2. List the storage classes in your cluster:

```
kubectl get storageclass
```

3. Remove the default designation from the default storage class. Replace <SC_NAME> with the name of the storage class:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"false"}}}'
```

4. Mark a different storage class as default. Replace <SC_NAME> with the name of the storage class:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

5. Confirm the new default storage class:

```
kubectl get storageclass
```

Add a private provider-managed cluster to Astra Control Service

You can use Astra Control Service to manage the following types of private provider-managed clusters:

- Amazon Elastic Kubernetes Service (EKS)
- Azure Kubernetes Service (AKS)
- Google Kubernetes Engine (GKE)
- Red Hat OpenShift Service on AWS (ROSA)
- ROSA with AWS PrivateLink

These instructions assume that you have already created a private cluster and prepared a secure method to remotely access it; for more information about creating and accessing private clusters, refer to the following documentation:

- Azure documentation for private AKS clusters
- Azure documentation for private OpenShift clusters
- Amazon EKS documentation
- Google Kubernetes Engine (GKE) documentation
- Red Hat OpenShift Service on AWS (ROSA) documentation

You need to perform the following tasks to add your private cluster to Astra Control Service:

1. Install Astra Connector
2. Set up persistent storage
3. Add the private provider-managed cluster to Astra Control Service

Install Astra Connector

Before you add a private cluster, you need to install Astra Connector on the cluster so that Astra Control can communicate with it. Refer to Install Astra Connector for private clusters for instructions.

Set up persistent storage

Configure persistent storage for the cluster. Refer to the Get Started documentation for more information about configuring persistent storage:

- Set up Microsoft Azure with Azure NetApp Files
- Set up Microsoft Azure with Azure managed disks
- Set up Amazon Web Services
- Set up Google Cloud

Add the private provider-managed cluster to Astra Control Service

You can now add the private cluster to Astra Control Service.

When you manage Azure Kubernetes Service and Google Kubernetes Engine clusters, note that you have two options for Astra Trident installation and lifecycle management:

- You can use Astra Control Service to automatically manage the lifecycle of Astra Trident. To do this, make sure that Astra Trident is not installed on the cluster that you want to manage with Astra Control Service. In this case, Astra Trident automatically installs Astra Trident when you begin managing the cluster, and Astra Trident upgrades are handled automatically.
- You can manage the lifecycle of Astra Trident yourself. To do this, install Astra Trident on the cluster before managing the cluster with Astra Control Service. In this case, Astra Control Service detects that Astra Trident is already installed and does not reinstall it or manage Astra Trident upgrades. Refer to the Astra Trident documentation for installation instructions.

When you manage Amazon Web Services clusters with Astra Control Service, if you need storage backends that are enabled by Astra Trident, you need to install Astra Trident manually on the cluster before you manage it with Astra Control Service. Refer to the Astra Trident documentation for installation instructions.

Before you begin

Amazon Web Services

- You should have the JSON file containing the credentials of the IAM user that created the cluster. Learn how to create an IAM user.
- Astra Trident is required for Amazon FSx for NetApp ONTAP. If you plan to use Amazon FSx for NetApp ONTAP as a storage backend for your EKS cluster, refer to the Astra Trident information in the EKS cluster requirements.
- (Optional) If you need to provide provide `kubectl` command access for a cluster to other IAM users that are not the cluster's creator, refer to the instructions in How do I provide access to other IAM users and roles after cluster creation in Amazon EKS?.
- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Amazon Web Services. Refer to the Cloud Volumes ONTAP setup documentation.

Microsoft Azure

- You should have the JSON file that contains the output from the Azure CLI when you created the service principal. Learn how to set up a service principal.

  You'll also need your Azure subscription ID, if you didn't add it to the JSON file.

- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Microsoft Azure. Refer to the Cloud Volumes ONTAP setup documentation.

Google Cloud

- You should have the service account key file for a service account that has the required permissions. Learn how to set up a service account.
- If the cluster is private, the authorized networks must allow the Astra Control Service IP address:

  52.188.218.166/32

- If you plan to use NetApp Cloud Volumes ONTAP as a storage backend, you need to configure Cloud Volumes ONTAP to work with Google Cloud. Refer to the Cloud Volumes ONTAP setup documentation.

Steps

1. (Optional) If you are adding an Amazon EKS cluster or want to manage the installation and upgrades of Astra Trident yourself, install Astra Trident on the cluster. Refer to the Astra Trident documentation for installation instructions.
2. Open the Astra Control Service web UI in a browser.
3. On the Dashboard, select **Manage Kubernetes cluster**.

   Follow the prompts to add the cluster.

4. **Provider**: Select your cloud provider and then either provide the required credentials to create a new cloud instance, or select an existing cloud instance to use.
   a. **Amazon Web Services**: Provide details about your Amazon Web Services IAM user account by uploading a JSON file or by pasting the contents of that JSON file from your clipboard.

The JSON file should contain the credentials of the IAM user that created the cluster.

b. **Microsoft Azure**: Provide details about your Azure service principal by uploading a JSON file or by pasting the contents of that JSON file from your clipboard.

   The JSON file should contain the output from the Azure CLI when you created the service principal. It can also include your subscription ID so it's automatically added to Astra. Otherwise, you need to manually enter the ID after providing the JSON.

c. **Google Cloud Platform**: Provide the service account key file either by uploading the file or by pasting the contents from your clipboard.

   Astra Control Service uses the service account to discover clusters running in Google Kubernetes Engine.

d. **Other**: This tab is for use with self-managed clusters only.

5. **Cloud instance name**: Provide a name for the new cloud instance that will be created when you add this cluster. Learn more about cloud instances.
6. Select **Next**.

   Astra Control Service displays a list of clusters that you can choose from.

7. **Cluster**: Select a cluster from the list to add to Astra Control Service.

   ⓘ When you are selecting from the list of clusters, pay careful attention to the **Eligiblity** column. If a cluster is "Ineligible" or "Partially eligible", hover over the status to determine if there's an issue with the cluster. For example, it might identify that the cluster doesn't have a worker node.

1. Select **Next**.
2. (Optional) **Storage**: Optionally, select the storage class that you'd like Kubernetes applications deployed to this cluster to use by default.
   a. To select a new default storage class for the cluster, enable the **Assign a new default storage class** check box.
   b. Select a new default storage class from the list.

> ⓘ Each cloud provider storage service displays the following price, performance, and resilience information:
>
> - Cloud Volumes Service for Google Cloud: Price, performance, and resilience information
> - Google Persistent Disk: No price, performance, or resilience information available
> - Azure NetApp Files: Performance and resilience information
> - Azure Managed disks: No price, performance, or resilience information available
> - Amazon Elastic Block Store: No price, performance, or resilience information available
> - Amazon FSx for NetApp ONTAP: No price, performance, or resilience information available
> - NetApp Cloud Volumes ONTAP: No price, performance, or resilience information available

Each storage class can utilize one of the following services:

- Cloud Volumes Service for Google Cloud
- Google Persistent Disk
- Azure NetApp Files
- Azure managed disks
- Amazon Elastic Block Store
- Amazon FSx for NetApp ONTAP
- NetApp Cloud Volumes ONTAP

  Learn more about storage classes for Amazon Web Services clusters, storage classes for GKE clusters, and storage classes for AKS clusters.

3. Select **Next**.
4. **Review & Approve**: Review the configuration details.
5. Select **Add** to add the cluster to Astra Control Service.

Result

If this is the first cluster that you have added for this cloud provider, Astra Control Service creates an object store for the cloud provider for backups of applications running on eligible clusters. (When you add subsequent clusters for this cloud provider, no further object stores are created.) If you specified a default storage class, Astra Control Service sets the default storage class that you specified. For clusters managed in Amazon Web Services or Google Cloud Platform, Astra Control Service also creates an admin account on the cluster. These actions can take several minutes.

Change the default storage class

You can change the default storage class for a cluster.

Change the default storage class using Astra Control

You can change the default storage class for a cluster from within Astra Control. If your cluster uses a previously installed storage backend service, you might not be able to use this method to change the default storage class (the **Set as default** action is not selectable). In this case, you can Change the default storage class using the command line.

Steps

1. In the Astra Control Service 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**.

Change the default storage class using the command line

You can change the default storage class for a cluster using Kubernetes commands. This method works regardless of your cluster's configuration.

Steps

1. Log in to your Kubernetes cluster.
2. List the storage classes in your cluster:

   ```
   kubectl get storageclass
   ```

3. Remove the default designation from the default storage class. Replace <SC_NAME> with the name of the storage class:

   ```
   kubectl patch storageclass <SC_NAME> -p '{"metadata":
   {"annotations":{"storageclass.kubernetes.io/is-default-
   class":"false"}}}'
   ```

4. Mark a different storage class as default. Replace <SC_NAME> with the name of the storage class:

   ```
   kubectl patch storageclass <SC_NAME> -p '{"metadata":
   {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
   ```

5. Confirm the new default storage class:

   ```
   kubectl get storageclass
   ```

Add a self-managed cluster

Add a public self-managed cluster to Astra Control Service

After you set up your environment, you’re ready to create a Kubernetes cluster and then add it to Astra Control Service.

A self-managed cluster is a cluster that you directly provision and manage. Astra Control Service supports self-managed clusters that run in a public cloud environment. You can add a self-managed cluster to Astra Control Service by uploading a `kubeconfig.yaml` file. You’ll need to ensure the cluster meets the requirements outlined here.

Supported Kubernetes distributions

You can use Astra Control Service to manage the following types of public, self-managed clusters:

Kubernetes distribution	Supported versions
Kubernetes (Upstream)	1.26 to 1.28 (with Astra Trident 23.04 or newer)
Rancher Kubernetes Engine (RKE)	RKE 1.3 with Rancher 2.6 RKE 1.4 with Rancher 2.7 RKE 2 (v1.23.x) with Rancher 2.6 RKE 2 (v1.24.x) with Rancher 2.7
Red Hat OpenShift Container Platform	4.11 through 4.14

These instructions assume that you have already created a self-managed cluster.

- Add the cluster to Astra Control Service
- Change the default storage class

Add the cluster to Astra Control Service

After you log in to Astra Control Service, your first step is to start managing your clusters. Before you add a cluster to Astra Control Service, you’ll need to perform specific tasks and make sure the cluster meets certain requirements.

Before you begin

A self-managed cluster is a cluster that you directly provision and manage. Astra Control Service supports self-managed clusters that run in a public cloud environment. Your self-managed clusters can use Astra Trident to interface with NetApp storage services, or they can use Container Storage Interface (CSI) drivers to interface with Amazon Elastic Block Store (EBS), Azure Managed Disks, and Google Persistent Disk.

Astra Control Service supports self-managed clusters that use the following Kubernetes distributions:

- Red Hat OpenShift Container Platform
- Rancher Kubernetes Engine
- Upstream Kubernetes

Your self-managed cluster needs to meet the following requirements:

- The cluster must be accessible via the internet.
- If you are using or plan to use storage enabled with CSI drivers, the appropriate CSI drivers must be installed on the cluster. For more information on using CSI drivers to integrate storage, refer to the documentation for your storage service.
- You have access to the cluster kubeconfig file that includes only one context element. Follow these instructions to generate a kubeconfig file.
- 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 Trident**: If you are using or plan to use NetApp storage, ensure that you have installed the latest version of Astra Trident. If Astra Trident is already installed, check to make sure it is the latest version.

  > You can deploy Astra Trident using either Trident operator (manually or using Helm chart) or `tridentctl`. Prior to installing or upgrading Astra Trident, review the supported frontends, backends, and host configurations.

  - **Astra Trident storage backend configured**: At least one Astra Trident storage backend must be configured on the cluster.
  - **Astra Trident storage classes configured**: At least one Astra Trident storage class must be configured on the cluster. If a default storage class is configured, ensure that only one storage class has that annotation.
  - **Astra Trident volume snapshot controller and volume snapshot class installed and configured**: The volume snapshot controller must be installed so that snapshots can be created in Astra Control. At least one Astra Trident `VolumeSnapshotClass` has been set up by an

administrator.

- **Astra Control Provisioner**: To use Astra Control Provisioner advanced management and storage provisioning features that are only accessible to Astra Control users, you must install Astra Trident 23.10 or later and enable Astra Control Provisioner functionality.

Steps

1. On the Dashboard, select **Manage Kubernetes cluster**.

   Follow the prompts to add the cluster.

2. **Provider**: Select the **Other** tab to add details about your self-managed cluster.

   a. **Other**: Provide details about your self-managed cluster by uploading a `kubeconfig.yaml` file or by pasting the contents of the `kubeconfig.yaml` file from your clipboard.

   > ⓘ 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.

3. **Credential name**: Provide a name for the self-managed cluster credential you are uploading to Astra Control. By default, the credential name is auto-populated as the name of the cluster.
4. **Private route identifier**: This field is for use with private clusters only.
5. Select **Next**.
6. (Optional) **Storage**: Optionally, select the storage class that you'd like Kubernetes applications deployed to this cluster to use by default.

   a. To select a new default storage class for the cluster, enable the **Assign a new default storage class** check box.

   b. Select a new default storage class from the list.

   > ⓘ Each cloud provider storage service displays the following price, performance, and resilience information:
   >
   > - Cloud Volumes Service for Google Cloud: Price, performance, and resilience information
   > - Google Persistent Disk: No price, performance, or resilience information available
   > - Azure NetApp Files: Performance and resilience information
   > - Azure Managed disks: No price, performance, or resilience information available
   > - Amazon Elastic Block Store: No price, performance, or resilience information available
   > - Amazon FSx for NetApp ONTAP: No price, performance, or resilience information available
   > - NetApp Cloud Volumes ONTAP: No price, performance, or resilience information available

   Each storage class can utilize one of the following services:

   - Cloud Volumes Service for Google Cloud

    - Google Persistent Disk
    - Azure NetApp Files
    - Azure managed disks
    - Amazon Elastic Block Store
    - Amazon FSx for NetApp ONTAP
    - NetApp Cloud Volumes ONTAP

    Learn more about storage classes for Amazon Web Services clusters, storage classes for GKE clusters, and storage classes for AKS clusters.

7. Select **Next**.
8. **Review & Approve**: Review the configuration details.
9. Select **Add** to add the cluster to Astra Control Service.

Change the default storage class

You can change the default storage class for a cluster.

Change the default storage class using Astra Control

You can change the default storage class for a cluster from within Astra Control. If your cluster uses a previously installed storage backend service, you might not be able to use this method to change the default storage class (the **Set as default** action is not selectable). In this case, you can Change the default storage class using the command line.

Steps

1. In the Astra Control Service 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**.

Change the default storage class using the command line

You can change the default storage class for a cluster using Kubernetes commands. This method works regardless of your cluster's configuration.

Steps

1. Log in to your Kubernetes cluster.
2. List the storage classes in your cluster:

    ```
    kubectl get storageclass
    ```

3. Remove the default designation from the default storage class. Replace <SC_NAME> with the name of the storage class:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-
class":"false"}}}'
```

4. Mark a different storage class as default. Replace <SC_NAME> with the name of the storage class:

```
kubectl patch storageclass <SC_NAME> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

5. Confirm the new default storage class:

```
kubectl get storageclass
```

Add a private self-managed cluster to Astra Control Service

After you set up your environment, you're ready to create a Kubernetes cluster and then add it to Astra Control Service.

A self-managed cluster is a cluster that you directly provision and manage. Astra Control Service supports self-managed clusters that run in a public cloud environment. You can add a self-managed cluster to Astra Control Service by uploading a `kubeconfig.yaml` file. You'll need to ensure the cluster meets the requirements outlined here.

Supported Kubernetes distributions

You can use Astra Control Service to manage the following types of private, self-managed clusters:

Kubernetes distribution	Supported versions
Kubernetes (Upstream)	1.26 to 1.28
Rancher Kubernetes Engine (RKE)	RKE 1.3 with Rancher Manager 2.6 RKE 1.4 with Rancher Manager 2.7 RKE 2 (v1.24.x) with Rancher 2.6 RKE 2 (v1.26.x) with Rancher 2.7
Red Hat OpenShift Container Platform	4.11 through 4.14

These instructions assume that you have already created a private cluster and prepared a secure method to remotely access it.

You need to perform the following tasks to add your private cluster to Astra Control Service:

1. Install Astra Connector
2. Set up persistent storage
3. Add the private self-managed cluster to Astra Control Service

Install Astra Connector

Before you add a private cluster, you need to install Astra Connector on the cluster so that Astra Control can communicate with it. Refer to Install Astra Connector for private clusters for instructions.

Set up persistent storage

Configure persistent storage for the cluster. Refer to the Get Started documentation for more information about configuring persistent storage:

- Set up Microsoft Azure with Azure NetApp Files
- Set up Microsoft Azure with Azure managed disks
- Set up Amazon Web Services
- Set up Google Cloud

Add the private self-managed cluster to Astra Control Service

You can now add the private cluster to Astra Control Service.

Before you begin

A self-managed cluster is a cluster that you directly provision and manage. Astra Control Service supports self-managed clusters that run in a public cloud environment. Your self-managed clusters can use Astra Trident to interface with NetApp storage services, or they can use Container Storage Interface (CSI) drivers to interface with Amazon Elastic Block Store (EBS), Azure Managed Disks, and Google Persistent Disk.

Astra Control Service supports self-managed clusters that use the following Kubernetes distributions:

- Red Hat OpenShift Container Platform
- Rancher Kubernetes Engine
- Upstream Kubernetes

Your self-managed cluster needs to meet the following requirements:

- The cluster must be accessible via the internet.
- If you are using or plan to use storage enabled with CSI drivers, the appropriate CSI drivers must be installed on the cluster. For more information on using CSI drivers to integrate storage, refer to the documentation for your storage service.
- You have access to the cluster kubeconfig file that includes only one context element. Follow these instructions to generate a kubeconfig file.
- 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 Trident**: If you are using or plan to use NetApp storage, ensure that you have installed the latest version of Astra Trident. If Astra Trident is already installed, check to make sure it is the latest version.

  > ⓘ You can deploy Astra Trident using either Trident operator (manually or using Helm chart) or `tridentctl`. Prior to installing or upgrading Astra Trident, review the supported frontends, backends, and host configurations.

  - **Astra Trident storage backend configured**: At least one Astra Trident storage backend must be configured on the cluster.
  - **Astra Trident storage classes configured**: At least one Astra Trident storage class must be configured on the cluster. If a default storage class is configured, ensure that only one storage class has that annotation.
  - **Astra Trident volume snapshot controller and volume snapshot class installed and configured**: The volume snapshot controller must be installed so that snapshots can be created in Astra Control. At least one Astra Trident `VolumeSnapshotClass` has been set up by an

administrator.

- **Astra Control Provisioner**: To use Astra Control Provisioner advanced management and storage provisioning features that are only accessible to Astra Control users, you must install Astra Trident 23.10 or later and enable Astra Control Provisioner functionality.

Steps

1. On the Dashboard, select **Manage Kubernetes cluster**.

   Follow the prompts to add the cluster.

2. **Provider**: Select the **Other** tab to add details about your self-managed cluster.
3. **Other**: Provide details about your self-managed cluster by uploading a `kubeconfig.yaml` file or by pasting the contents of the `kubeconfig.yaml` file from your clipboard.

   > If you create your own `kubeconfig` file, you should define only **one** context element in it. Refer to these instructions for information about creating `kubeconfig` files.

4. **Credential name**: Provide a name for the self-managed cluster credential you are uploading to Astra Control. By default, the credential name is auto-populated as the name of the cluster.
5. **Private route identifier**: Enter the private route identifier, which you can obtain from the Astra Connector. If you query the Astra Connector via the `kubectl get astraconnector -n astra-connector` command, the private route identifier is referred to as the `ASTRACONNECTORID`.

   > The private route identifier is the name associated with the Astra Connector that enables a private Kubernetes cluster to be managed by Astra. In this context, a private cluster is a Kubernetes cluster that does not expose its API server to the internet.

6. Select **Next**.
7. (Optional) **Storage**: Optionally, select the storage class that you’d like Kubernetes applications deployed to this cluster to use by default.
   1. To select a new default storage class for the cluster, enable the **Assign a new default storage class** check box.
   2. Select a new default storage class from the list.

ⓘ Each cloud provider storage service displays the following price, performance, and resilience information:

- Cloud Volumes Service for Google Cloud: Price, performance, and resilience information
- Google Persistent Disk: No price, performance, or resilience information available
- Azure NetApp Files: Performance and resilience information
- Azure Managed disks: No price, performance, or resilience information available
- Amazon Elastic Block Store: No price, performance, or resilience information available
- Amazon FSx for NetApp ONTAP: No price, performance, or resilience information available
- NetApp Cloud Volumes ONTAP: No price, performance, or resilience information available

Each storage class can utilize one of the following services:

- Cloud Volumes Service for Google Cloud
- Google Persistent Disk
- Azure NetApp Files
- Azure managed disks
- Amazon Elastic Block Store
- Amazon FSx for NetApp ONTAP
- NetApp Cloud Volumes ONTAP

  Learn more about storage classes for Amazon Web Services clusters, storage classes for GKE clusters, and storage classes for AKS clusters.

8. Select **Next**.
9. **Review & Approve**: Review the configuration details.
10. Select **Add** to add the cluster to Astra Control Service.

Change the default storage class

You can change the default storage class for a cluster.

Change the default storage class using Astra Control

You can change the default storage class for a cluster from within Astra Control. If your cluster uses a previously installed storage backend service, you might not be able to use this method to change the default storage class (the **Set as default** action is not selectable). In this case, you can Change the default storage class using the command line.

Steps

1. In the Astra Control Service 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**.

Change the default storage class using the command line

You can change the default storage class for a cluster using Kubernetes commands. This method works regardless of your cluster’s configuration.

Steps

1. Log in to your Kubernetes cluster.
2. List the storage classes in your cluster:

   ```
   kubectl get storageclass
   ```

3. Remove the default designation from the default storage class. Replace <SC_NAME> with the name of the storage class:

   ```
   kubectl patch storageclass <SC_NAME> -p '{"metadata":
   {"annotations":{"storageclass.kubernetes.io/is-default-
   class":"false"}}}'
   ```

4. Mark a different storage class as default. Replace <SC_NAME> with the name of the storage class:

   ```
   kubectl patch storageclass <SC_NAME> -p '{"metadata":
   {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
   ```

5. Confirm the new default storage class:

   ```
   kubectl get storageclass
   ```

Check the Astra Trident version

To add a self-managed cluster that uses Astra Trident for storage services, ensure that the installed version of Astra Trident is the latest.

Steps

1. Check the Astra Trident version.

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

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

```
NAME      VERSION
trident   22.10.0
```

If Astra Trident is not installed, you see output similar to the following:

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

> If Astra Trident is not installed or not current, and you want your cluster to use Astra Trident for storage services, you need to install the latest version of Astra Trident before proceeding. Refer to the Astra Trident documentation for instructions.

2. Ensure that the pods are running:

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

3. Check if the storage classes are using the supported Astra Trident drivers. The provisioner name should be `csi.trident.netapp.io`. Refer to 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 kubeconfig file

You can add a cluster to Astra Control Service using a kubeconfig file. Depending on the type of cluster you want to add, you might need to manually create a kubeconfig file for your cluster using specific steps.

- Create a kubeconfig file for Amazon EKS clusters
- Create a kubeconfig file for Red Hat OpenShift Service on AWS (ROSA) clusters
- Create a kubeconfig file for other types of clusters

Create a kubeconfig file for Amazon EKS clusters

Follow these instructions to create a kubeconfig file and permanent token secret for Amazon EKS clusters. A permanent token secret is required for clusters hosted in EKS.

Steps

1. Follow the instructions in the Amazon documentation to generate a kubeconfig file:

   Creating or updating a kubeconfig file for an Amazon EKS cluster

2. Create a service account as follows:

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

      Adjust the service account name as needed. The namespace `kube-system` is required for these steps. If you change the service account name here, you should apply the same changes in the following steps.

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

      ```
      apiVersion: v1
      kind: ServiceAccount
      metadata:
        name: astra-admin-account
        namespace: kube-system
      ```

3. Apply the service account:

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

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

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

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRoleBinding
   metadata:
     name: astra-admin-binding
   roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: ClusterRole
     name: cluster-admin
   subjects:
   - kind: ServiceAccount
     name: astra-admin-account
     namespace: kube-system
   ```

5. Apply the cluster role binding:

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

6. Create a service account token secret file called `astracontrol-secret.yaml`.

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

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: astra-admin-account
  name: astra-admin-account
  namespace: kube-system
type: kubernetes.io/service-account-token
```

7. Apply the token secret:

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

8. Retrieve the token secret:

```
kubectl get secret astra-admin-account -n kube-system -o
jsonpath='{.data.token}' | base64 -d
```

9. Replace the `user` section of the AWS EKS kubeconfig file with the token, as shown in the following example:

```
user:
    token: k8s-aws-
v1.aHR0cHM6Ly9zdHMudXMtd2VzdC0yLmFtYXpvbmF3cy5jb20vP0FjdGlvbj1HZXRDYWxsZ
XJJZGVudGl0eSZWZXJzaW9uPTIwMTEtMDYtMTUmWC1BbXotQWxnb3JpdGhtPUFXUzQtSE1BQ
y1TSEEyNTYmWC1BbXotQ3JlZGVudGlhbD1BS0lBM1JEWDdKU0haWU9LSEQ2SyUyRjIwMjMwN
DAzJTJGdXMtd2VzdC0yJTJGc3RzJTJGYXdzNF9yZXF1ZXN0JlgtQW16LURhdGU9MjAyMzA0M
DNUMjA0MzQwWiZYLUFtei1FeHBpcmVzPTYwJlgtQW16LVNpZ25lZEhlYWRlcnM9aG9zdCUzQ
ngtazhzLWF3cy1pZCZYLUFtei1TaWduYXR1cmU9YjU4ZWM0NzdiM2NkZGYxNGRhNzU4MGI2Z
WQ2zY2NzI2YWIwM2UyNThjMjRhNTJjNmVhNjc4MTRlNjJkOTg2Mg
```

Create a kubeconfig file for Red Hat OpenShift Service on AWS (ROSA) clusters

Follow these instructions to create a kubeconfig file for Red Hat OpenShift Service on AWS (ROSA) clusters.

Steps

1. Log in to the ROSA cluster.
2. Create a service account:

   ```
   oc create sa astracontrol-service-account
   ```

3. Add a cluster role:

   ```
   oc adm policy add-cluster-role-to-user cluster-admin -z astracontrol-
   service-account
   ```

4. Using the following example, create a service account secret configuration file:

   ```
   <strong>secret-astra-sa.yaml</strong>
   ```

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

5. Create the secret:

   ```
   oc create -f secret-astra-sa.yaml
   ```

6. Edit the service account that you created, and add the Astra Control service account secret name to the `secrets` section:

   ```
   oc edit sa astracontrol-service-account
   ```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-dvfcd
kind: ServiceAccount
metadata:
  creationTimestamp: "2023-08-04T04:18:30Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "169770"
  uid: 965fa151-923f-4fbd-9289-30cad15998ac
secrets:
- name: astracontrol-service-account-dockercfg-dvfcd
- name: secret-astracontrol-service-account ####ADD THIS ONLY####
```

7. 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-dvfcd"},
   { "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-dvfcd` 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 will need this index number in the next step.

8. Generate the kubeconfig as follows:

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

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

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

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

CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

Create a kubeconfig file for other types of clusters

Follow these instructions to create a kubeconfig file for Rancher, Upstream Kubernetes, and Red Hat OpenShift clusters.

Before you begin

Ensure that you have the following on your machine before you start:

- kubectl v1.26 or later installed
- 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`.

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

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

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

   b. Apply the service account:

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

2. 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:

Expand for steps

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

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

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

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

   # Justification for resource permissions:

   # Astra Control needs to be able to discover (list) resources
   of all types within your application.
   # These permissions are required to discover, back up, and
   restore your application resources including
   # secrets.
   # For example, if your application contains custom resources
   or cluster-scoped resources, Astra Control
   # needs '*' to discover, back up, and restore your application
   resources.

   # Justification for Verbs:
   # - "List" enables discovery.
   # - "Get" enables resource backups and enables users to define
   apps using GVK.
   # - "Create" enables restoring an application from a snapshot
   or backup using Astra Control.
   # - "Delete" enables application resource clean-up as part of
   an in-place restore of an application or clones.
   # - "Patch" enables maintaining owner references and updating
   labels on some resources.
   # - "Update" enables replica scaling in case of operations
   like in-place restores of your application.
   # - "Watch" enables Astra Control to keep an up to date view
   of resources.

   # Manage all resources
   # Necessary to back up and restore all resources in an app
   - apiGroups:
   ```

```
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch
  - delete
  - watch
  - update

- nonResourceURLs:
  - /metrics
  verbs:
  - get
  - watch
  - list

# Use PodSecurityPolicies
- apiGroups:
  - extensions
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use

# OpenShift security - uncomment the following lines for Red Hat OpenShift clusters
#- apiGroups:
#  - security.openshift.io
#  resources:
#  - securitycontextconstraints
#  verbs:
#  - use
```

b. (For OpenShift clusters only) If you are creating a kubeconfig for an OpenShift cluster, uncomment the final lines in the `astra-admin-account.yaml` file after the `# Use PodSecurityPolicies` section:

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

c. 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.

Expand for steps

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

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

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

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

b. 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`.

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

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

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: 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 will need this index number in the next step.

7. Generate the kubeconfig as follows:

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

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

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

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

CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

What's next?

Now that you’ve logged in and added a cluster to Astra Control, you’re ready to start using Astra Control’s application data management features.

- Start managing apps
- Protect apps
- Clone apps
- Set up billing
- Invite and manage users
- Manage cloud provider credentials
- Manage notifications

Astra Control Service videos

Check out NetApp TV for the latest video content featuring Astra Control Service. NetApp TV includes videos that demonstrate certain features of Astra Control Service or show you how to complete certain common tasks.

Astra Control Service videos

Frequently asked questions for Astra Control Service

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

Overview

Astra Control aims to simplify your application data lifecycle management operations for Kubernetes native applications. Astra Control Service supports Kubernetes clusters running on multiple cloud provider environments.

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

Access to Astra Control

Why do I need to provide so many details when registering for Astra Control?

Astra Control requires accurate customer information when registering. This information is required to go through a Global Trade Compliance (GTC) check.

Why am I getting a “Registration Failed” error when registering for Astra Control?

Astra Control requires you to provide accurate customer information in the onboarding section. You will get a "Registration Failed" error if you provided incorrect information. Other accounts that you are a member of also get locked.

What’s the Astra Control Service URL?

You can access Astra Control Service at https://astra.netapp.io.

I sent an email invitation to a colleague, but they haven’t received it. What should I do?

Ask them to check their spam folder for an email from do-not-reply@netapp.com, or search their inbox for "invitation." You can also remove the user and attempt to re-add them.

I upgraded to the Premium PayGO Plan from the Free Plan. Will I get charged for the first 10 namespaces?

Yes. After upgrading to the Premium Plan, Astra Control starts charging you for all managed namespaces in your account.

I upgraded to the Premium PayGO Plan in the middle of a month. Will I get charged for the entire month?

No. Billing starts from the time that you upgraded to the Premium Plan.

I am using the Free Plan, will I get charged for the Persistent Volume Claims?

Yes, you will be charged for the persistent volumes used by the clusters from your cloud provider.

Registering Kubernetes clusters

Do I need to install CSI drivers on my cluster before adding it to Astra Control Service?

No. When your cluster is added to Astra Control, the service will automatically install NetApp's Trident Container Storage Interface (CSI) driver on the Kubernetes cluster. This CSI driver is used to provision persistent volumes for clusters backed by your cloud provider.

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

New worker nodes can be added to existing pools, or new pools can be created as long as they are the `COS_CONTAINERD` image type. 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.

Registering Elastic Kubernetes Service (EKS) clusters

Can I add a private EKS cluster to Astra Control Service?

Yes, you can add private EKS clusters to Astra Control Service. To add a private EKS cluster, refer to Start managing Kubernetes clusters from Astra Control Service.

Registering Azure Kubernetes Service (AKS) clusters

Can I add a private AKS cluster to Astra Control Service?

Yes, you can add private AKS clusters to Astra Control Service. To add a private AKS cluster, refer to Start managing Kubernetes clusters from Astra Control Service.

Can I use Active Directory to manage authentication for my AKS clusters?

Yes, you can configure your AKS clusters to use Azure Active Directory (Azure AD) for authentication and identity management. When you create the cluster, follow the instructions in the official documentation to configure the cluster to use Azure AD. You'll need to make sure your clusters meet the requirements for AKS-managed Azure AD integration.

Registering Google Kubernetes Engine (GKE) clusters

Can I add a private GKE cluster to Astra Control Service?

Yes, you can add private GKE clusters to Astra Control Service. To add a private GKE cluster, refer to Start managing Kubernetes clusters from Astra Control Service.

Private GKE clusters must have the authorized networks set to allow the Astra Control IP address:

52.188.218.166/32

Can my GKE cluster reside on a shared VPC?

Yes. Astra Control can manage clusters that reside in a shared VPC. Learn how to set up the Astra service account for a shared VPC configuration.

Where can I find my service account credentials on GCP?

After you log in to the Google Cloud Console, your service account details will be in the **IAM and Admin** section. For more details, refer to how to set up Google Cloud for Astra Control.

I would like to add different GKE clusters from different GCP projects. Is this supported in Astra Control?

No, this isn’t a supported configuration. Only a single GCP project is supported.

Removing clusters

How do I properly unregister, bring down a cluster, and delete the associated volumes?

1. Unmanage the applications from Astra Control.
2. Unregister the cluster from Astra Control.
3. Delete the persistent volume claims.
4. Delete the cluster.

What happens to my applications and data after removing the 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. Volume snapshot data stored within the storage backend will not be removed. Persistent Storage backups created by Astra Control will remain within your cloud provider’s object store, but they are unavailable for restore.

> Always remove a cluster from Astra Control before you delete it through GCP. Deleting a cluster from GCP while it’s still being managed by Astra Control can cause problems for your Astra Control account.

Will Astra Trident be uninstalled when I remove a cluster from Astra Control?

Astra Trident will not be uninstalled from a cluster when you remove the cluster from Astra Control.

Managing applications

Can Astra Control deploy an application?

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

I don’t see any of my application’s PVCs bound to GCP CVS. What’s wrong?

The Astra Trident operator sets the default storage class to `netapp-cvs-perf-premium` after it’s successfully added to Astra Control. When an application’s PVCs are not bound to Cloud Volumes Service for Google Cloud, there are a few steps that you can take:

- Run `kubectl get sc` and check the default storage class.
- Check the yaml file or Helm chart that was used to deploy the application and see if a different storage class is defined.
- GKE version 1.24 and later does not support Docker-based node images. Check to make sure that the worker node image type in GKE is `COS_CONTAINERD` and that the NFS mount succeeded.

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.

Data management operations

Where does Astra Control create the object store bucket?

The geography of the first managed cluster determines the location of the object store. For example, if the first cluster that you add is in a European zone, then the bucket is created in that same geography. If needed, you can add additional buckets.

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

In some situations, Astra Control will automatically create a snapshot as part of performing another process. If these snapshots are more than a few minutes old, you can safely delete them.

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

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

Can I manage snapshots taken by Astra Control directly through my cloud provider?

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?

In coming Astra Control updates, Astra Control Provisioner will replace Astra Trident as storage provisioner and orchestrator in the Astra Control architecture. Because of this, it’s highly 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.

Do I have to pay for Astra Trident?

No. Astra Trident will continue to be open source and free to download.

Can I use the storage management and provisioning features in Astra Control without installing and using all of Astra Control?

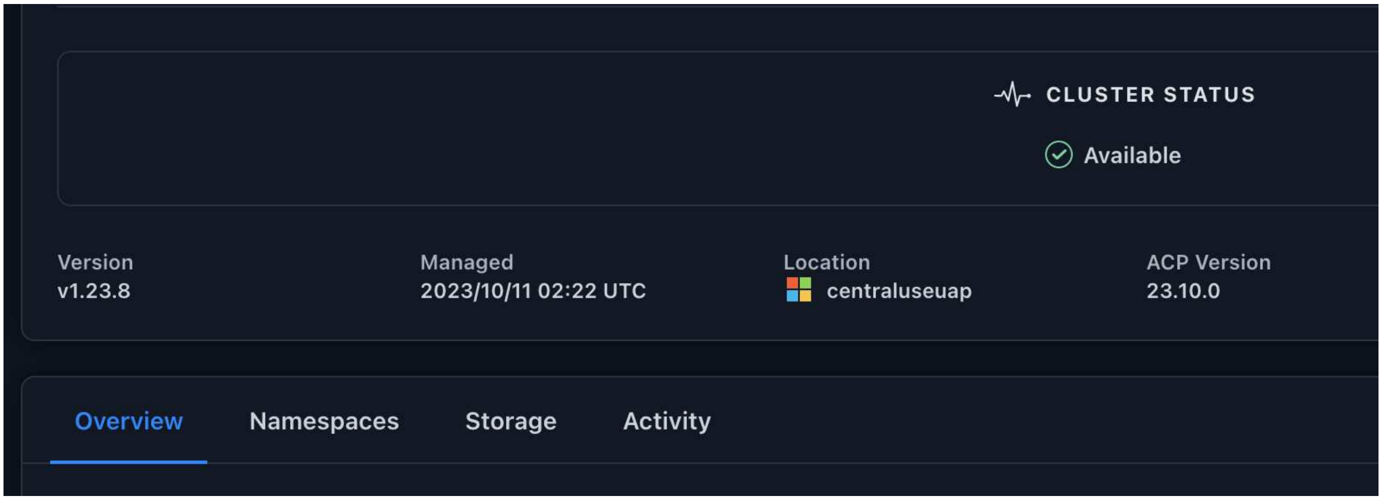
Yes, you can upgrade to Astra Trident 23.10 or later and enable Astra Control Provisioner functionality even if you do not want to consume the complete feature set of Astra Control data management functionality.

How can I transition from being an existing Trident user to Astra Control to use advanced storage management and provisioning functionality?

If you are an existing 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.



CLUSTER STATUS

Available

Version
v1.23.8

Managed
2023/10/11 02:22 UTC

Location
centraluseuap

ACP Version
23.10.0

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` with a status of `Installed`:

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

Response:

```
status:
  acpVersion: 23.10.0
  currentInstallationParams:
    ...
    acpImage: <my_custom_registry>/trident-acp:v23.10.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

tridentctl

Confirm that Astra Control Provisioner has been enabled:

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

Response:

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 23.10.0 | 23.10.0 | 23.10.0. | +----------------
+----------------+-------------+
```

Copyright information

Trademark information