# Install Astra Connector to manage clusters

## Astra Control Service

NetApp
June 11, 2024

# Table of Contents

# Install Astra Connector to manage clusters

Astra Connector is software that resides on your managed clusters and facilitates communication between the managed cluster and Astra Control. For clusters managed using Astra Control Service, there are two available versions of Astra Connector:

- **Previous version of Astra Connector**: Install the previous version of Astra Connector on your cluster if you plan to manage the cluster with non-Kubernetes-native workflows.
- [Tech preview] **Declarative Kubernetes Astra Connector**: Install Astra Connector for clusters managed with declarative Kubernetes workflows on your cluster if you plan to manage the cluster using declarative Kubernetes workflows. After you install the Astra Connector on your cluster, the cluster is automatically added to Astra Control.

The declarative Kubernetes Astra Connector is available only as part of the Astra Control Early Adopter Program (EAP). Contact your NetApp sales representative for information about joining the EAP.

# Install the previous version of Astra Connector

Astra Control Service uses the previous version of Astra Connector to enable communication between Astra Control Service and private clusters managed with non-Kubernetes-native workflows. You need to install Astra Connector on private clusters that you want to manage with non-Kubernetes-native workflows.

The previous version of Astra Connector supports the following types of private clusters managed with non-Kubernetes-native workflows:

- 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

**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 previous Astra Connector operator on the private cluster you want to manage with non-Kubernetes-native workflows. 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

# (Tech preview) Install the declarative Kubernetes Astra Connector

Clusters managed using declarative Kubernetes workflows use Astra Connector to enable communication between the managed cluster and Astra Control. You need to install Astra Connector on all clusters that you will manage with declarative Kubernetes workflows.

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

### About this task

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

### Before you begin

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

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

### Steps

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

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

2. Verify that the operator is installed and ready:

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

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

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

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

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

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

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

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

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

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

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

8. Verify that the Astra Connector is fully deployed:

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

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

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

You should see output similar to the following:

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

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

## Copyright information



## Trademark information