Get started

Astra Control Center

NetApp
February 12, 2024

Table of Contents

Get started

Astra Control Center requirements

Get started by verifying support for your Kubernetes clusters, apps, licenses, and web browser.

Kubernetes cluster general requirements

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

- **Image registry**: You must have an existing private Docker image registry to which you can push Astra Control Center build images. You must have the URL of the image registry where you will upload the images, and you must have tagged the images for the private container registry.
- **Trident / ONTAP storage configuration**: Astra Control Center requires that Trident version 21.01 or 21.04 already be installed and configured to work with NetApp ONTAP version 9.5 or newer as the storage backend. Astra Control Center requires that a storage class be created and set as the default storage class. Astra Control Center supports the following ONTAP drivers provided by Trident:
  - ontap-nas
  - ontap-nas-flexgroup
  - ontap-san
  - ontap-san-economy

If you are planning to manage the Kubernetes cluster from Astra Control Center as well as use the cluster to host the Astra Control Center installation, the cluster has the following additional requirements:

- The most recent version of the Kubernetes snapshot-controller component is installed
- A Trident volumesnapshotclass object has been defined by an administrator
- A default Kubernetes storage class exists on the cluster
- At least one storage class is configured to use Trident
- A method for pointing the FQDN of Astra Control Center to the external IP address of the Astra Control Center service

OpenShift clusters

Astra Control Center requires a Red Hat OpenShift Container Platform 4.6.8 or 4.7 cluster that has Trident storage classes backed by ONTAP 9.5 or newer, with the following attributes:

- At least 300GB of available ONTAP storage capacity
- 3 controller nodes with 4 CPU cores, 16GB RAM, and 120GB of available storage each
- 3 worker nodes with at least 12 CPU cores, 32GB RAM, and 50GB of available storage each
- Kubernetes version 1.19 or 1.20
- Service type "LoadBalancer" available for ingress traffic to be sent to services in the OpenShift cluster
- A method for pointing the FQDN of Astra Control Center to the load balanced IP address

These minimum requirements assume that Astra Control Center is the only application running on the OpenShift cluster. If the cluster is running additional applications, you need to adjust these minimum requirements accordingly.

Make sure that your cluster meets the minimum requirements and that you follow Kubernetes best practices so that Astra Control Center is highly available in your Kubernetes cluster.

OpenShift 4.8 is not supported.

During app cloning, Astra Control Center needs to allow OpenShift to mount volumes and change the ownership of files. Because of this, ONTAP needs to be configured to allow volume operations to complete successfully using the following commands:

1. `export-policy rule modify -vserver svm0 -policyname default -ruleindex 1 -superuser sys`
2. `export-policy rule modify -policyname default -ruleindex 1 -anon 65534`

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

App management requirements

Astra Control Center has the following app management requirements:

- **Licensing**: You need an Astra Control Center license to manage apps using Astra Control Center.
- **Helm 3**: If you use Helm to deploy apps, Astra Control Center requires Helm version 3. Managing and cloning apps deployed with Helm 3 (or upgraded from Helm 2 to Helm 3) are fully supported. Apps deployed with Helm 2 are not supported.
- **Operator management**: Astra Control Center does not support apps that are deployed with Operator Lifecycle Manager (OLM)-enabled operators or cluster-scoped operators.

Access to the internet

You should determine whether you have outside access to the internet. If you do not, some functionality might be limited, such as receiving monitoring and metrics data from NetApp Cloud Insights, or sending support bundles to the NetApp Support Site.

License

Astra Control Center requires an Astra Control Center license for full functionality. Obtain an evaluation license or full license from NetApp. Without a license, you will be unable to:

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

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

Service type "LoadBalancer" for on-premises Kubernetes clusters

Astra Control Center uses a service of the type "LoadBalancer" (svc/traefik in the Astra Control Center namespace), and requires that it be assigned an accessible external IP address. For on-premises OpenShift clusters, you can use MetalLB to automatically assign an external IP address to the service. In the internal DNS server configuration, you should point the chosen DNS name for Astra Control Center to the load-balanced IP address.

Networking requirements

The cluster that hosts Astra Control Center communicates using the following TCP ports. You should ensure that these ports are allowed through any firewalls, and configure firewalls to allow any HTTPS egress traffic originating from the Astra network. Some ports require connectivity both ways between the cluster hosting Astra Control Center and each managed cluster (noted where applicable).

Product	Port	Protocol	Direction	Purpose
Astra Control Center	443	HTTPS	Ingress	UI / API access - Ensure this port is open both ways between the cluster hosting Astra Control Center and each managed cluster
Astra Control Center	9090	HTTPS	• Ingress (to cluster hosting Astra Control Center) • Egress (random port from the node IP address of each worker node of each managed cluster)	Metrics data to metrics consumer - ensure each managed cluster can access this port on the cluster hosting Astra Control Center
Trident	34571	HTTPS	Ingress	Node pod communication
Trident	9220	HTTP	Ingress	Metrics endpoint

Supported web browsers

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

What's next

View the quick start overview.

Quick start for Astra Control Center

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

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

Review Kubernetes cluster requirements

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

Learn more about the Astra Control Center requirements.

Download and install Astra Control Center

- Download Astra Control Center from the NetApp Support Site.
- Install Astra Control Center in your local environment.
- Discover your Trident configuration backed by the ONTAP storage backend.

  For our first release, you'll install the images on an OpenShift registry or use your local registry.

Learn more about installing Astra Control Center.

Complete some initial setup tasks

- Add a license.
- Add a Kubernetes cluster and Astra Control Center discovers details.
- Add an ONTAP storage backend.
- Optionally, add an object store bucket that will store your app backups.

Learn more about the initial setup process.

Use Astra Control Center

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

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

Continue from this Quick Start

Install Astra Control Center.

Find more information

- Use the Astra API

Install Astra Control Center

To install Astra Control Center, do the following steps:

- Install Astra Control Center
- Log in to the Astra Control Center UI

Install Astra Control Center

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

What you'll need

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

```
oc get clusteroperators
```

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

```
oc get apiservices
```

About this task

The Astra Control Center installation process does the following:

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

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

Steps

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

2. Download the zip of Astra Control Center certificates and keys from NetApp Support Site.
3. (Optional) Use the following command to verify the signature of the bundle:

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

4. Extract the images:

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

5. Change to the Astra directory.

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

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

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

   a. Log in to your Docker registry:

   ```
   docker login [Docker_registry_path]
   ```

   b. Load the images into Docker.

   c. Tag the images.

   d. Push the images to your local registry.

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

7. (For registries with auth requirements only) If you use a registry that requires authentication, you need to do the following:

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

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

Response:

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

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

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

Sample response:

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

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

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

Sample response:

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

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

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

Response

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

8. Edit the Astra Control Center operator deployment yaml

(`astra_control_center_operator_deploy.yaml`) to refer to your local registry and secret.

```
vim astra_control_center_operator_deploy.yaml
```

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

```
imagePullSecrets:
- name: astra-registry-cred
```

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

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: [Docker_registry_path]/kube-rbac-proxy:v0.5.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [Docker_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

9. Edit the Astra Control Center custom resource (CR) file (`astra_control_center_min.yaml`):

```
vim astra_control_center_min.yaml
```

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

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

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

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

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

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

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

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

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

h. If you are not using a registry that requires authorization, delete the `secret` line.

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

10. Install the Astra Control Center operator:

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

Sample response:

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

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

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

Sample response:

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

12. Run the following patch to correct cluster role binding.
13. Install Astra Control Center in the `netapp-acc` (or your custom) namespace:

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

Sample response:

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

14. Verify that all system components installed successfully.

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

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

Sample response:

```
NAME                                      READY   STATUS    RESTARTS
AGE
acc-helm-repo-5fdfff786f-gkv6z            1/1     Running   0
4m58s
activity-649f869bf7-jn5gs                 1/1     Running   0
3m14s
asup-79846b5fdc-s9s97                     1/1     Running   0
3m10s
authentication-84c78f5cf4-qhx9t           1/1     Running   0
118s
billing-9b8496787-v8rzv                   1/1     Running   0
2m54s
bucketservice-5fb876d9d5-wkfvz            1/1     Running   0
3m26s
cloud-extension-f9f4f59c6-dz6s6           1/1     Running   0
3m
cloud-insights-service-5676b8c6d4-6q7lv   1/1     Running   0
2m52s
composite-compute-7dcc9c6d6c-lxdr6        1/1     Running   0
2m50s
composite-volume-74dbfd7577-cd42b         1/1     Running   0
3m2s
credentials-75dbf46f9d-5qm2b              1/1     Running   0
3m32s
entitlement-6cf875cb48-gkvhp              1/1     Running   0
3m12s
features-74fd97bb46-vss2n                 1/1     Running   0
3m6s
fluent-bit-ds-2g9jb                       1/1     Running   0
113s
fluent-bit-ds-5tg5h                       1/1     Running   0
113s
fluent-bit-ds-qfxb8                       1/1     Running   0
113s
graphql-server-7769f98b86-p4qrv           1/1     Running   0
90s
identity-566c566cd5-ntfj6                 1/1     Running   0
3m16s
```

```
influxdb2-0                                  1/1     Running   0
4m43s
krakend-5cb8d56978-44q66                     1/1     Running   0
93s
license-66cbbc6f48-27kgf                     1/1     Running   0
3m4s
login-ui-584f7fd84b-dmdrp                    1/1     Running   0
87s
loki-0                                       1/1     Running   0
4m44s
metrics-ingestion-service-6dcfddf45f-mhnvh   1/1     Running   0
3m8s
monitoring-operator-78d67b4d4-nxs6v          2/2     Running   0
116s
nats-0                                       1/1     Running   0
4m40s
nats-1                                       1/1     Running   0
4m26s
nats-2                                       1/1     Running   0
4m15s
nautilus-9b664bc55-rn9t8                     1/1     Running   0
2m56s
openapi-dc5ddfb7d-6q8vh                      1/1     Running   0
3m20s
polaris-consul-consul-5tjs7                  1/1     Running   0
4m43s
polaris-consul-consul-5wbnx                  1/1     Running   0
4m43s
polaris-consul-consul-bfvl7                  1/1     Running   0
4m43s
polaris-consul-consul-server-0               1/1     Running   0
4m43s
polaris-consul-consul-server-1               1/1     Running   0
4m43s
polaris-consul-consul-server-2               1/1     Running   0
4m43s
polaris-mongodb-0                            2/2     Running   0
4m49s
polaris-mongodb-1                            2/2     Running   0
4m22s
polaris-mongodb-arbiter-0                    1/1     Running   0
4m49s
polaris-ui-6648875998-75d98                  1/1     Running   0
92s
polaris-vault-0                              1/1     Running   0
4m41s
```

```
polaris-vault-1                              1/1     Running   0
4m41s
polaris-vault-2                              1/1     Running   0
4m41s
storage-backend-metrics-69546f4fc8-m7lfj     1/1     Running   0
3m22s
storage-provider-5d46f755b-qfv89             1/1     Running   0
3m30s
support-5dc579865c-z4pwq                     1/1     Running   0
3m18s
telegraf-ds-4452f                            1/1     Running   0
113s
telegraf-ds-gnqxl                            1/1     Running   0
113s
telegraf-ds-jhw74                            1/1     Running   0
113s
telegraf-rs-gg6m4                            1/1     Running   0
113s
telemetry-service-6dcc875f98-zft26           1/1     Running   0
3m24s
tenancy-7f7f77f699-q7l6w                     1/1     Running   0
3m28s
traefik-769d846f9b-c9crt                     1/1     Running   0
83s
traefik-769d846f9b-l9n4k                     1/1     Running   0
67s
trident-svc-8649c8bfc5-pdj79                 1/1     Running   0
2m57s
vault-controller-745879f98b-49c5v            1/1     Running   0
4m51s
```

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

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

16. When all the pods are running, verify installation success by retrieving the AstraControlCenter instance installed by the ACC Operator.

```
kubectl get acc -o yaml -n netapp-acc
```

17. Check the `status.deploymentState` field in the response for the `Deployed` value. If deployment was unsuccessful, an error message appears instead.

You will use the `uuid` in the next step.

```
apiVersion: v1
items:
- apiVersion: astra.netapp.io/v1
  kind: AstraControlCenter
  metadata:
    creationTimestamp: "2021-07-28T21:36:49Z"
    finalizers:
    - astracontrolcenter.netapp.io/finalizer
   generation: 1
    name: astra
    namespace: netapp-acc
    resourceVersion: "27797604"
    selfLink: /apis/astra.netapp.io/v1/namespaces/netapp-
acc/astracontrolcenters/astra
    uid: 61cd8b65-047b-431a-ba35-510afcb845f1
  spec:
    accountName: Example
    astraAddress: astra.example.com
    astraResourcesScaler: "Off"
    astraVersion: 21.08.52
    autoSupport:
      enrolled: false
    email: admin@example.com
    firstName: SRE
    lastName: Admin
    imageRegistry:
      name: registry_name/astra
  status:
    certManager: deploy
    deploymentState: Deployed
    observedGeneration: 1
    observedVersion: 21.08.52
    postInstall: Complete
    uuid: c49008a5-4ef1-4c5d-a53e-830daf994116
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

18. To get the one-time password you will use when you log in to Astra Control Center, copy the `status.uuid` value from the response in the previous step. The password is `ACC-` followed by the UUID value (`ACC-[UUID]` or, in this example, `ACC-c49008a5-4ef1-4c5d-a53e-830daf994116`).

Log in to the Astra Control Center UI

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

Steps

1. In a browser, enter the FQDN you used in the `astraAddress` in the `astra_control_center_min.yaml` CR when you installed ACC.
2. Accept the self-signed certificates when prompted.

   You can create a custom certificate after login.

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

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

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

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

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

Troubleshoot the installation

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

Steps

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

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

What's next

Complete the deployment by performing setup tasks.

Set up Astra Control Center

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

Tasks

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

Add a license for Astra Control Center

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

What you'll need

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

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

Add a full or evaluation license

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

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

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

Steps

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

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

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

Add cluster

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

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

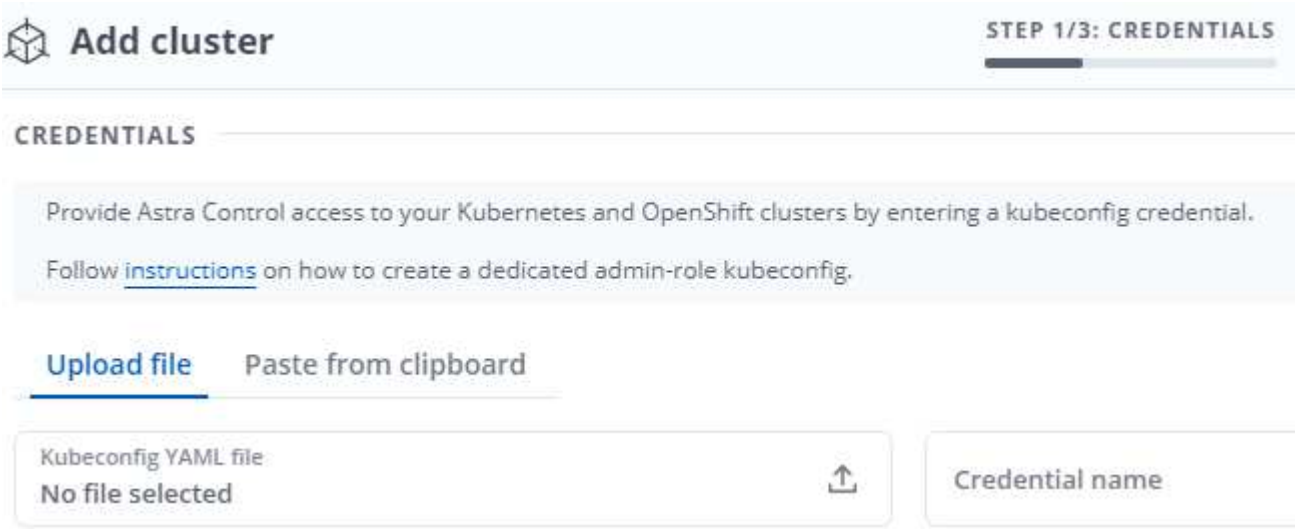
What you'll need

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

Steps

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

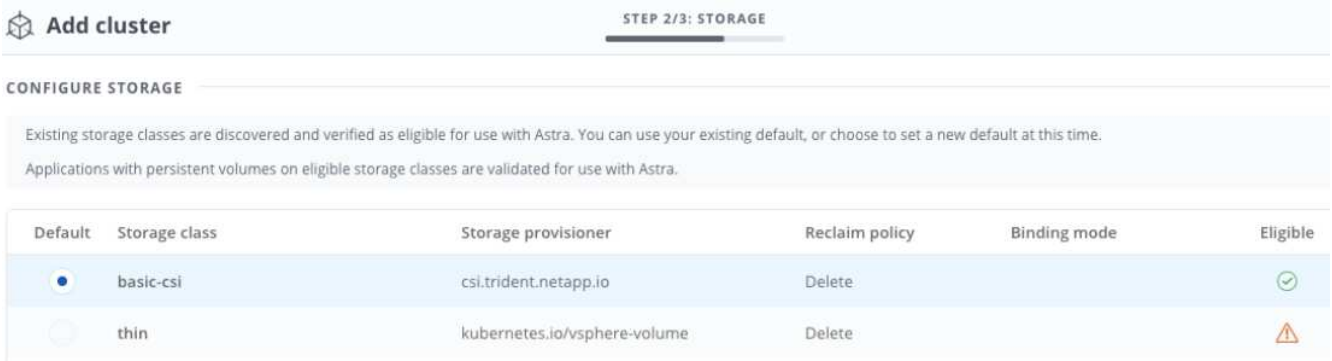
   The `kubeconfig.yaml` file should include **only the cluster credential for one cluster**.



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

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

   You should select a Trident storage class backed by ONTAP storage.



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

Result

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

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

Add a storage backend

You can add a storage backend so that Astra Control can manage its resources. Managing storage clusters in Astra Control as a storage backend enables you to get linkages between persistent volumes (PVs) and the storage backend as well as additional storage metrics.

You can add a storage backend in the following ways:

- Configure storage when you are adding a cluster. See Add cluster.
- Add a discovered storage backend using either the Dashboard or the Backends option.

You can add an already discovered storage backend using these options:

- Add storage backend using Dashboard
- Add storage backend using Backends option

Add storage backend using Dashboard

1. From the Dashboard do one of the following:
   a. From the Dashboard Storage backend section, select **Manage**.
   b. From the Dashboard Resource Summary > Storage backends section, select **Add**.
2. Enter the ONTAP admin credentials and select **Review**.
3. Confirm the backend details and select **Manage**.

   The backend appears in the list with summary information.

Add storage backend using Backends option

1. In the left navigation area, select **Backends**.
2. Select **Manage**.
3. Enter the ONTAP admin credentials and select **Review**.
4. Confirm the backend details and select **Manage**.

   The backend appears in the list with summary information.

5. To see details of the backend storage, select it.

   Persistent volumes used by apps in the managed compute cluster are also displayed.

Add a bucket

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

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

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

Use any of the following bucket types:

- NetApp ONTAP S3
- NetApp StorageGRID S3
- Generic S3

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

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

Steps

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

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

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

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

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

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

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

f. Continue by adding credential information.

Add S3 access credentials

Add S3 access credentials at any time.

Steps

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

What's next?

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

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

Find more information

- Use the Astra API
- Known issues

Prerequisites for adding a cluster

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

What you'll need before you add a cluster

- A cluster running OpenShift 4.6 or 4.7, which has Trident StorageClasses backed by ONTAP 9.5 or later.

- One or more worker nodes with at least 1GB RAM available for running telemetry services.

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

- The superuser and user ID set on the backing ONTAP system to back up and restore apps with Astra Control Center (ACC). Run the following commands in the ONTAP command line:
`export policy rule modify -vserver svm0 -policyname default -ruleindex 1 -superuser sys`
`export-policy rule modify -policyname default -ruleindex 1 -anon 65534` (this is the default value)

Run eligibility checks

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

Steps

1. Check the Trident version.

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

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

```
NAME      VERSION
trident   21.04.0
```

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

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

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

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

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

Create an admin-role kubeconfig

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

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

Steps

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

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

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

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

   b. Apply the service account:

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

2. Grant cluster admin permissions as follows:
   a. Create a `ClusterRoleBinding` file called `astracontrol-clusterrolebinding.yaml`.

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

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

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

b. Apply the cluster role binding:

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

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

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

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

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

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

4. Generate the kubeconfig as follows:

a. Create a `create-kubeconfig.sh` file. If the token index you noted in the previous step was not 0, replace the value for `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 the value for
```

```
TOKEN_INDEX from
# the output in the previous step if it was not 0. 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'
TOKEN_INDEX=0

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
```

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

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

What's next?

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

Find more information

- Trident documentation
- Use the Astra API

Add a custom TLS certificate

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

What you'll need

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

Remove the self-signed certificate

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

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

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

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

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

Add a new certificate

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

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

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

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

#spec:
#  selfSigned: {}

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

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

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

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

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

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

5. Create the certificate using the following command:

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

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

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

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

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

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

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

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

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

Frequently asked questions for Astra Control Center

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

Overview

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

Access to Astra Control Center

What's the Astra Control URL?

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

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

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

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

Steps

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

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

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

Registering Kubernetes clusters

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

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

How do I properly unmanage a cluster?

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

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

Removing a cluster from Astra Control will not make any changes to the cluster's configuration (applications and persistent storage). Any Astra Control snapshots or backups taken of applications on that cluster will be unavailable to restore. Persistent storage backups created by Astra Control remain within Astra Control, but they are unavailable for restore.

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

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

Trident will not be uninstalled from a cluster when you remove it 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.

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

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

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

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

Should I manage Astra Control itself?

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

Data management operations

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

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

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

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

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

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

Copyright information

Trademark information