



# **Set up Astra Control Center**

## **Astra Control Center**

NetApp

November 15, 2021

This PDF was generated from <https://docs.netapp.com/us-en/astra-control-center/get-started/add-cluster-reqs.html> on November 15, 2021. Always check docs.netapp.com for the latest.

# Table of Contents

- Set up Astra Control Center ..... 1
  - Add a license for Astra Control Center. .... 1
  - Add cluster ..... 2
  - Add a storage backend ..... 3
  - Add a bucket ..... 4
  - What's next?..... 5
  - Prerequisites for adding a cluster. .... 5
  - Add a custom TLS certificate ..... 10

# 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**.



## Add cluster

STEP 1/3: CREDENTIALS

### CREDENTIALS

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

Upload file

Paste from clipboard

Kubeconfig YAML file  
No file selected



Credential name



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

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



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

## CONFIGURE STORAGE

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

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

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

**Result**

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



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

## Add a storage backend

You can add a storage backend so that Astra Control can manage its resources. 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.



## Copyright Information

Copyright © 2021 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

## Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.