



Set up Astra Control Center

Astra Control Center

NetApp

March 18, 2024

This PDF was generated from <https://docs.netapp.com/us-en/astra-control-center/get-started/add-license.html> on March 18, 2024. 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
 - Enable Astra Control Provisioner 1
 - Prepare your environment for cluster management using Astra Control 11
 - (Tech preview) Install Astra Connector for managed clusters 23
 - Add a cluster 26
 - Enable authentication on an ONTAP storage backend 27
 - Add a storage backend 33
 - Add a bucket 34

Set up Astra Control Center

Add a license for Astra Control Center

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

You can add a new license using the Astra Control UI or [Astra Control API](#).

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



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



To update an existing evaluation or full license, refer to [Update an existing license](#).

Before you begin

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

Steps

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

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



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

Enable Astra Control Provisioner

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

In coming Astra Control updates, Astra Control Provisioner will replace Astra Trident as storage provisioner and orchestrator and be mandatory for Astra Control use. Because of this, it's strongly recommended that Astra Control users enable Astra Control Provisioner. Astra Trident will continue to remain open source and be

released, maintained, supported, and updated with new CSI and other features from NetApp.

About this task

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

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

Before you begin

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

Astra Control Provisioners users with Astra Control Center

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

Astra Control Provisioner only users

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

(Step 1) Get the Astra Control Provisioner image

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

Astra Control image registry



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

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

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

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

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



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

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

Example:

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

- b. Tag the image:

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

- c. Push the image to your custom registry:

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



You can use Crane copy as an alternative to running these Docker commands:

```
crane copy cr.astra.netapp.io/trident-acp:24.02.0  
<my_custom_registry>/trident-acp:24.02.0
```

NetApp Support Site

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

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

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

3. Load the Astra Control Provisioner image:

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

Response:

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

4. Tag the image:

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

5. Push the image to your custom registry:

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

(Step 2) Enable Astra Control Provisioner in Astra Trident

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

Astra Trident operator

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

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

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



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

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

4. Verify Astra Trident is running:

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

Response:

NAME	AGE
trident	21m

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

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

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

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

- a. Set a custom registry location for the Astra Trident image or pull it from the Astra Control registry (tridentImage: <my_custom_registry>/trident:24.02.0 or tridentImage: netapp/trident:24.02.0).
- b. Enable Astra Control Provisioner (enableACP: true).
- c. Set the custom registry location for the Astra Control Provisioner image or pull it from the Astra Control registry (acpImage: <my_custom_registry>/trident-acp:24.02.0 or acpImage: cr.astra.netapp.io/trident-acp:24.02.0).
- d. If you established [image pull secrets](#) earlier in this procedure, you can set them here (imagePullSecrets: - <secret_name>). Use the same name secret name you established in the previous steps.

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

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

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



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

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

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

Response:

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

tridentctl

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

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

4. Confirm that Astra Control Provisioner has been enabled:

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

Response:

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

Helm

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

```
kubectl delete psp tridentoperatorpod
```

3. Add the Astra Trident Helm repository:

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

4. Update the Helm chart:

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

Response:

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

5. List the images:

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

Response:

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

6. Ensure that trident-operator 24.02.0 is available:

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

Response:

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

7. Use `helm install` and run one of the following options that include these settings:

- A name for your deployment location
- The Astra Trident version
- The name of the Astra Control Provisioner image
- The flag to enable the provisioner
- (Optional) A local registry path. If you are using a local registry, your [Trident images](#) can be located in one registry or different registries, but all CSI images must be located in the same registry.
- The Trident namespace

Options

- Images without a registry

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

- Images in one or more registries

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

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



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

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

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

Result

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

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

CLUSTER STATUS

Available

Version v1.24.9+rke2r2	Managed 2024/03/15 17:32 UTC	Kube-system namespace UID <div></div>	ACP Version <div></div>
Private route identifier <div>...</div>	Cloud instance private	Default bucket astra-bucket1 (inherited)	

Overview

Namespaces

Storage

Activity

For more information

- [Astra Trident upgrades documentation](#)

Prepare your environment for cluster management using Astra Control

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

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

Before you begin

- **Meet environmental prerequisites:** Your environment meets [operational environment requirements](#) for Astra Control Center.
- **Configure worker nodes:** Ensure that you [configure the worker nodes](#) in your cluster with the appropriate

storage drivers so that the pods can interact with the backend storage.

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

- netapp-acc-operator namespace:

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

- netapp monitoring namespace:

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

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

Run the following commands in the ONTAP command line:

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

- **kubeconfig-managed cluster requirements:** These requirements are specific for app clusters managed by kubeconfig.

- **Make kubeconfig accessible:** You have access to the [default cluster kubeconfig](#) that [you configured during installation](#).
- **Certificate Authority considerations:** If you are adding the cluster using a kubeconfig file that references a private Certificate Authority (CA), add the following line to the `cluster` section of the kubeconfig file. This enables Astra Control to add the cluster:

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

- **Rancher only:** When managing application clusters in a Rancher environment, modify the application cluster's default context in the kubeconfig file provided by Rancher to use a control plane context instead of the Rancher API server context. This reduces load on the Rancher API server and improves performance.
- **Astra Control Provisioner requirements:** You should have a properly configured Astra Control Provisioner, including its Astra Trident components, to manage clusters.
 - **Review Astra Trident environment requirements:** Prior to installing or upgrading Astra Control Provisioner, review the [supported frontends, backends, and host configurations](#).

- **Enable Astra Control Provisioner functionality:** It's highly recommended that you install Astra Trident 23.10 or later and enable [Astra Control Provisioner advanced storage functionality](#). In coming releases, Astra Control will not support Astra Trident if the Astra Control Provisioner is not also enabled.
- **Configure a storage backend:** At least one storage backend must be [configured in Astra Trident](#) on the cluster.
- **Configure a storage class:** At least one storage class must be [configured in Astra Trident](#) on the cluster. If a default storage class is configured, ensure that it is the **only** storage class that has the default annotation.
- **Configure a volume snapshot controller and install a volume snapshot class:** [Install a volume snapshot controller](#) so that snapshots can be created in Astra Control. [Create](#) at least one `VolumeSnapshotClass` using Astra Trident.

Run eligibility checks

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

Steps

1. Determine the Astra Trident version you are running:

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

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

NAME	VERSION
trident	24.02.0

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

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

2. Do one of the following:

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

3. Ensure that all pods (including `trident-acp`) are running:

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

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

```
kubectl get sc
```

Sample response:

NAME	PROVISIONER	RECLAIMPOLICY
ontap-gold (default)	csi.trident.netapp.io	Delete
VolumeBindingMode: AllowVolumeExpansion: true	5d23h	Immediate

Create a cluster role kubeconfig

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

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

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

Before you begin

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

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



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

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

Steps

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

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

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

b. Apply the service account:

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

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

Limited cluster role

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

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

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

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

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

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

```

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

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

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

```

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

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

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

3. Apply the cluster role:

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

Expanded cluster role

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



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

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

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

```

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

```

2. Apply the cluster role:

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

3. Create the cluster role binding for the cluster role to the service account:

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

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

```

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

```

b. Apply the cluster role binding:

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

4. Create and apply the token secret:

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

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

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

- b. Apply the token secret:

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

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

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

```

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

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

```

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

```

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

```

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

```

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

```

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

7. Generate the kubeconfig as follows:

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

```

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

```

```

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

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

CONTEXT=$(kubectl config current-context)

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

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

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

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

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

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

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

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

```

```

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

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

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

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

```

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

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

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

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(Tech preview) Install Astra Connector for managed clusters

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

Install Astra Connector

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

About this task

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

Before you begin

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



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

Steps

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

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

2. Verify that the operator is installed and ready:

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

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

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

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



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

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

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

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

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

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

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

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

8. Verify that the Astra Connector is fully deployed:

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

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

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

You should see output similar to the following:

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

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

Add a cluster

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



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

Before you begin

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

Steps

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



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



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

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



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

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

Result

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



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

Enable authentication on an ONTAP storage backend

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

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

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

Enable credential-based authentication

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



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

A sample backend definition looks like this:

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

The backend definition is the only place the credentials are stored in plain text. The creation or update of a

backend is the only step that requires knowledge of the credentials. As such, it is an admin-only operation, to be performed by the Kubernetes or storage administrator.

Enable certificate-based authentication

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

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

You can use one of the following types of certificates:

- Self-signed certificate
- Third-party certificate

Enable authentication with a self-signed certificate

A typical workflow involves the following steps.

Steps

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

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

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

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

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

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

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

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

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

Enable authentication with a third-party certificate

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

Steps

1. Generate the private key and CSR:

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

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

You now have the following three files:

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

Expand for sample.yaml

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

```
security certificate install -type server-ca
```

```
Please enter Certificate: Press <Enter> when done
```

```
-----BEGIN CERTIFICATE-----
```

```
<certificate details>
```

```
-----END CERTIFICATE-----
```

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

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

CA:

serial:

The certificate's generated name for reference:

===

```
# Copy the contents of ontap_signed_cert.crt and use it here. For key, use the contents of ontap_cert_request.key file.
```

```
security certificate install -type server
```

```
Please enter Certificate: Press <Enter> when done
```

```
-----BEGIN CERTIFICATE-----
```

```
<certificate details>
```

```
-----END CERTIFICATE-----
```

```
Please enter Private Key: Press <Enter> when done
```

```
-----BEGIN PRIVATE KEY-----
```

```
<private key details>
```

```
-----END PRIVATE KEY-----
```

Enter certificates of certification authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate.

Do you want to continue entering root and/or intermediate

```
certificates {y|n}: n
```

The provided certificate does not have a common name in the subject field.

Enter a valid common name to continue installation of the certificate: <ONTAP_CLUSTER_FQDN_NAME>

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

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

CA:

serial:

The certificate's generated name for reference:

```
==
```

```
# Modify the vsrver settings to enable SSL for the installed
certificate
```

```
ssl modify -vsrver <vsrver_name> -ca <CA> -server-enabled true
-serial <serial number>          (security ssl modify)
```

```
==
```

```
# Verify if the certificate works fine:
```

```
openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
```

```
CONNECTED(00000005)
```

```
depth=1 DC = local, DC = umca, CN = <CA>
```

```
verify return:1
```

```
depth=0
```

```
verify return:1
```

```
write W BLOCK
```

```
---
```

```
Certificate chain
```

```
0 s:
```

```
    i:/DC=local/DC=umca/<CA>
```

```
-----BEGIN CERTIFICATE-----
```

```
<Certificate details>
```

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

Expand for sample.yaml

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

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

Please enter Certificate: Press <Enter> when done

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

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

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

==

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

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

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

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```

"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-4bf5378b41bd"
}
}
},
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%

```

9. Add the storage backend in the Astra Control Center UI and provide the following values:

- **Client Certificate:** ontap_test_client.pem
- **Private Key:** ontap_test_client.key
- **Trusted CA Certificate:** ontap_signed_cert.crt

Add a storage backend

After you set up the credentials or certificate authentication information, you can add an existing ONTAP storage backend to Astra Control Center to manage its resources.

Managing storage clusters in Astra Control as a storage backend enables you to get linkages between persistent volumes (PVs) and the storage backend as well as additional storage metrics.

Adding and managing ONTAP storage backends in Astra Control Center is optional when using NetApp SnapMirror technology if you have enabled Astra Control Provisioner.

Steps

1. From the Dashboard in the left-navigation area, select **Backends**.
2. Select **Add**.
3. In the Use Existing section of the Add storage backend page, select **ONTAP**.
4. Select one of the following:
 - **Use administrator credentials:** Enter the ONTAP cluster management IP address and admin credentials. The credentials must be cluster-wide credentials.



The user whose credentials you enter here must have the `ontapi` user login access method enabled within ONTAP System Manager on the ONTAP cluster. If you plan to use SnapMirror replication, apply user credentials with the "admin" role, which has the access methods `ontapi` and `http`, on both source and destination ONTAP clusters. Refer to [Manage User Accounts in ONTAP documentation](#) for more information.

- **Use a certificate:** Upload the certificate `.pem` file, the certificate key `.key` file, and optionally the certificate authority file.
5. Select **Next**.
 6. Confirm the backend details and select **Manage**.

Result

The backend appears in the `online` state in the list with summary information.



You might need to refresh the page for the backend to appear.

Add a bucket

You can add a bucket using the Astra Control UI or [Astra Control API](#). Adding object store bucket providers is essential if you want to back up your applications and persistent storage or if you want to clone applications across clusters. Astra Control stores those backups or clones in the object store buckets that you define.

You don't need a bucket in Astra Control if you are cloning your application configuration and persistent storage to the same cluster. Application snapshots functionality does not require a bucket.

Before you begin

- Ensure you have a bucket that is reachable from your clusters managed by Astra Control Center.
- Ensure you have credentials for the bucket.
- Ensure the bucket is one of the following types:
 - NetApp ONTAP S3
 - NetApp StorageGRID S3
 - Microsoft Azure
 - Generic S3



Amazon Web Services (AWS) and Google Cloud Platform (GCP) use the Generic S3 bucket type.



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

Steps

1. In the left navigation area, select **Buckets**.
2. Select **Add**.
3. Select the bucket type.



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

4. Enter an existing bucket name and optional description.



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

5. Enter the name or IP address of the S3 endpoint.
6. Under **Select Credentials**, choose either the **Add** or **Use existing** tab.
 - If you chose **Add**:
 - a. Enter a name for the credential that distinguishes it from other credentials in Astra Control.
 - b. Enter the access ID and secret key by pasting the contents from your clipboard.
 - If you chose **Use existing**:
 - a. Select the existing credentials you want to use with the bucket.
7. Select **Add**.



When you add a bucket, Astra Control marks one bucket with the default bucket indicator. The first bucket that you create becomes the default bucket. As you add buckets, you can later decide to [set another default bucket](#).

Copyright information

Copyright © 2024 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.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

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.