



Get started

Astra Data Store

NetApp
July 19, 2022

This PDF was generated from <https://docs.netapp.com/us-en/astra-data-store/get-started/requirements.html> on July 19, 2022. Always check docs.netapp.com for the latest.

Table of Contents

- Get started 1
 - Astra Data Store requirements 1
 - Quick start for Astra Data Store 5
 - Install Astra Data Store 6
 - Set up Astra Data Store components 24
 - Astra Data Store Early Access Program (EAP) release limits 35
 - Frequently asked questions for Astra Data Store 36

Get started

Astra Data Store requirements

Get started by verifying that your environment meets Astra Data Store requirements.

Astra Data Store supports both bare-metal and VM-based deployments. The Astra Data Store cluster can run on a Kubernetes cluster with four or more worker nodes. Astra Data Store software can co-exist with other applications running on the same Kubernetes cluster.

- [Kubernetes worker node resources](#)
- [Hardware and software](#)
- [Networking](#)
- [Astra Trident](#)
- [Container Network Interfaces](#)
- [Licensing](#)



If you plan to manage your Astra Data Store cluster from Astra Control Center, make sure your Astra Data Store cluster meets the [requirements for clusters that will be managed by Astra Control Center](#) in addition to the requirements outlined here.

Kubernetes worker node resources

These are the resource requirements required to be assigned to the Astra Data Store software on each worker node in the Kubernetes cluster:

Resource	Minimum	Maximum
Number of nodes in Astra Data Store cluster	4	16
Number of data drives	<ul style="list-style-type: none">• 3 (with a separate cache device present)• 4 (if there is no cache device present)	14
Data drive size	100GiB	4TiB
Number of optional cache devices	1 (8GiB in size)	N/A

Astra Data Store supports the following vCPU and RAM combinations for each Kubernetes worker node:

- 9 vCPUs with 38GiB of RAM
- 23 vCPUs with 94 GiB of RAM



For the best write performance, you should configure a dedicated high-endurance, low-latency, low-capacity cache device.

Each worker node has the following additional requirements:

- 100GiB or greater free space on host disk (boot) for Astra Data Store log files to be stored.
- At least one 10GbE or faster network interface for cluster, data, and management traffic. Optionally, an additional 1GbE or faster interface can be used to separate out management traffic.

Hardware and software

Astra Data Store software is validated on the following hardware platforms, software, and storage configuration. Visit [NetApp community support](#) if your Kubernetes cluster configuration is different.

Hardware platforms

- Dell R640
- Dell R740
- HPE DL360
- HPE DL380
- Lenovo SR630
- Lenovo SR650



VM-based deployments can also use servers that are listed as compatible on the [VMware Compatibility Guide](#).

Storage

Astra Data Store has been validated with SATA, SAS, and NVMe TLC SSDs.

For VM-based deployments, you can use drives presented as virtual disks or passthrough drives.



- If your host uses SSDs behind a hardware RAID controller, configure the hardware RAID controller to use "passthrough" mode.
- Each drive should have a unique serial number. Add the attribute `disk.enableuuid=TRUE` in the virtual machine advanced settings during VM creation.

Software

- Hypervisor: Astra Data Store is validated with VMware-based VM deployments with vSphere 7.0. KVM-based deployments are not supported by Astra Data Store.
- Astra Data Store has been validated on the following host operating systems:
 - Red Hat Enterprise Linux 8.4
 - Red Hat Enterprise Linux 8.2
 - Red Hat Enterprise Linux 7.9
 - Red Hat Enterprise Linux CoreOS (RHCOS)
 - CentOS 8
 - Ubuntu 20.04
- Astra Data Store has been validated with the following Kubernetes distributions:
 - Red Hat OpenShift 4.6 through 4.9

- Google Anthos 1.8 through 1.10
- Kubernetes 1.20 through 1.23
- Rancher RKE 1.3.3



Astra Data Store requires Astra Trident for storage provisioning and orchestration. Astra Trident versions 21.10.1 through 22.04 are supported. See the [Astra Trident installation instructions](#).

Networking

Astra Data Store requires one IP address per cluster for MVIP. It needs to be an unused or unconfigured IP address in same subnet as MIP. The Astra Data Store management interface should be same as the Kubernetes node's management interface.

In addition, each node can be configured as described in the following table:



The following abbreviations are used in this table:

MIP: Management IP address
CIP: Cluster IP address
MVIP: Management virtual IP address

Configuration	IP addresses needed
One network interface per node	<ul style="list-style-type: none"> • Two (2) per node: <ul style="list-style-type: none"> ◦ MIP/CIP: One (1) pre-configured IP address on management interface per node ◦ Data IP: One (1) unused or unconfigured IP address per node in same subnet as MIP
Two network interfaces per node	<ul style="list-style-type: none"> • Three per node: <ul style="list-style-type: none"> ◦ MIP: One (1) pre-configured IP address on management interface per node ◦ CIP: One (1) pre-configured IP address on data interface per node in a different subnet from MIP ◦ Data IP: One (1) unused or unconfigured IP address per node in same subnet as CIP



No VLAN tags are used in these configurations.

Firewall considerations

In environments that enforce network firewall traffic filtering, the firewall must be configured to allow incoming traffic to the ports and protocols listed in the table below. The IP Address column uses the following abbreviations:

- MIP: Primary IP address on the management interface of each node
- CIP: Primary IP address on the cluster interface of each node

- DIP: One or more data IP addresses configured on a node
- MVIP: The management virtual IP address configured on one cluster node


Port/protocol	IP address	Purpose	Notes
111/TCP	DIP	NFS	Data IPs move between cluster nodes at runtime. Each node should allow this traffic for all data IPs (or for the entire subnet).
442/TCP	MIP	API	
635/TCP	DIP	NFS	Data IPs move between cluster nodes at runtime. Each node should allow this traffic for all data IPs (or for the entire subnet).
2010/UDP	CIP	Cluster/node discovery	Includes both unicast and broadcast traffic to and from UDP port 2010, including the replies.
2049/TCP	DIP	NFS	Data IPs move between cluster nodes at runtime. Each node should allow this traffic for all data IPs (or for the entire subnet).
2181-2183/TCP	CIP	Distributed communication	
2443/TCP	MIP	API	
2443/TCP	MVIP	API	The MVIP address can be hosted by any cluster node and is relocated at runtime when needed.
4000-4006/TCP	CIP	Intra-cluster RPC	
4045-4046/TCP	DIP	NFS	Data IPs move between cluster nodes at runtime. Each node should allow this traffic for all data IPs (or for the entire subnet).
7700/TCP	CIP	Session manager	
9919/TCP	MIP	DMS API	
9920/TCP	DIP	DMS REST server	
ICMP	CIP + DIP	Intra-node communication, health check	Data IPs move between cluster nodes at runtime. Each node should allow this traffic for all data IPs (or for the entire subnet).

Astra Trident

Astra Data Store requires the application Kubernetes clusters to be running Astra Trident for storage provisioning and orchestration. Astra Trident versions 21.10.1 through 22.04 are supported. Astra Data Store can be configured as a [storage backend](#) with Astra Trident to provision persistent volumes.

Container Network Interfaces

Astra Data Store has been validated with the following Container Network Interfaces (CNIs).

- Calico for RKE clusters
 - Calico and Weave Net CNIs for vanilla Kubernetes clusters
 - OpenShift SDN for Red Hat OpenShift Container Platform (OCP)
 - Cilium for Google Anthos
- 
 - Astra Data Store deployed with the Cilium CNI requires the portmap plugin for HostPort support. You can enable CNI chaining mode by adding `cni-chaining-mode: portmap` to the cilium-config configMap and restarting the Cilium pods.
 - Firewall-enabled configurations are supported only for the Calico and OpenShift SDN CNIs.

Licensing

Astra Data Store requires a valid license to enable full functionality.

[Sign up here](#) to obtain the Astra Data Store license. Instructions to download the license will be sent to you after you sign up.

What's next

View the [quick start](#) overview.

For more information

[Astra Data Store limits](#)

Quick start for Astra Data Store

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

Try it out! If you want to try Astra Data Store, you can use a 90-day evaluation license. [Sign up here](#) to obtain an Astra Data Store license.

1

Review Kubernetes cluster requirements

- Clusters must be running in a healthy state, with at least four or more worker nodes.
- Each of the Kubernetes worker nodes that are part of the Astra Data Store deployment should have SSDs of the same interface type (SATA, SAS or NVMe) and the same number of drives that are assigned to the Astra Data Store cluster.

- Each SSD should have a unique serial number.

Learn more about [Astra Data Store requirements](#).

2

Download and install Astra Data Store

- Download Astra Data Store from the [NetApp Support Site](#).
- Install Astra Data Store in your local environment.
- Apply the Astra Data Store license.
- Install the Astra Data Store cluster.

Learn more about [installing Astra Data Store](#).

3

Complete some initial setup tasks

- Install Astra Trident.
- Install Kubernetes snapshot custom resource definitions (CRDs) and controller.
- Set up Astra Data Store as a storage backend.
- Create a default Astra Data Store storage class.
- Configure Astra Data Store for telemetry services.

Learn more about the [initial setup process](#).

4

Use Astra Data Store

After you finish setting up Astra Data Store, here's what you might do next:

- Use `kubectl` commands and `kubectl astrads` extension to manage the cluster, including tasks such as place a node in maintenance mode, replace a drive, or replace a node. Learn more about [how to use kubectl commands with Astra Data Store](#).
- Configure monitoring endpoints. Learn more about [configuring monitoring endpoints](#).
- [Use Astra Data Store with VMware](#).

5

Continue from this Quick Start

[Install Astra Data Store](#).

Install Astra Data Store

You can install Astra Data Store using Kubernetes-native commands or using the UI in Astra Control Center.

Installation options

- **With Kubernetes-native commands:** To install Astra Data Store using Kubernetes-native commands, complete the installation steps described in [this procedure](#).

- **With Astra Control Center:** To install Astra Data Store using Astra Control Center, complete the installation steps described in [this procedure](#).

What you'll need

- [Before you begin installation, prepare your environment for Astra Data Store deployment.](#)
- Access to the [NetApp Support Site](#). [Register](#) for an evaluation download if you don't already have a full-access NetApp Support Site account.
- A [NetApp license file \(NLF\)](#) for Astra Data Store. Instructions to download the license will be sent to you after you [sign up](#).
- An active kubeconfig with cluster admin rights for the active context.
- An understanding of [the roles and privileges](#) used by Astra Data Store.

Install Astra Data Store

The Astra Data Store installation process for standard Kubernetes clusters guides you through the following high-level steps. Additional installation steps for Red Hat OpenShift Container Platform (OCP) environments are also described.

- [Download the Astra Data Store bundle and extract the images](#)
- [Copy the binary and push images to your local registry](#)
- [OpenShift procedure](#)
- [Install the Astra Data Store operator](#)
- [Deploy the Astra Data Store version YAML](#)
- [Apply the Astra Data Store license](#)
- [Install the Astra Data Store cluster](#)
- [Understand deployment-related events](#)



To use Astra Data Store with Google Anthos, complete these installation steps and add the Astra Data Store cluster to your Anthos environment.



To install Astra Data Store with Rancher RKE environments, complete these installation steps and substitute Rancher commands for kubectl commands.

Download the Astra Data Store bundle and extract the images

1. Log in to the [NetApp Support Site](#) and download the Astra Data Store bundle (Astra_Data_Store_2022.05.01.tar).



If you are looking for instructions for earlier versions of the bundle, see [documentation for that version](#).

2. (Optional) Use the following command to verify the signature of the bundle:

```
openssl dgst -sha256 -verify Astra_Data_Store_2022.05.01.pub -signature  
Astra_Data_Store_2022.05.01.sig 2022.12.01_ads.tar
```

3. Create a directory:

```
mkdir Astra_Data_Store_2022.05.01  
cd Astra_Data_Store_2022.05.01
```

4. Extract the images:

```
tar -xvf <path to tar file>/Astra_Data_Store_2022.05.01.tar
```



The images will be extracted to a `astrads/images` directory created within the working directory.

Copy the binary and push images to your local registry

1. Copy the `kubectl-astrads` binary from the directory you used to extract images to the standard path where k8s `kubectl` binaries are installed (`/usr/bin/` is used as the path in the example below). `kubectl-astrads` is a custom `kubectl` extension that installs and manages Astra Data Store clusters.



Use the `which kubectl` command to find the path where `kubectl` binaries are installed.

```
cp -p ./astrads/bin/kubectl-astrads /usr/bin/.
```

2. Add the files in the Astra Data Store image directory to your local registry.



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

- a. Log in to your registry:

```
docker login [your_registry_path]
```

- b. Set an environment variable to the registry path where you want to push the Astra Data Store images; for example, `repo.company.com`.

```
export REGISTRY=repo.company.com/astrads
```

- c. Run the script to load the images into Docker, tag the images, and push the images to your local registry:

```
for astraImageFile in $(ls astrads/images/*.tar) ; do
    astraImage=$(docker load --input ${astraImageFile} | sed 's~Loaded
image: ~~')
    astraImageShort=`echo $astraImage | sed 's~.*/~~'`
    docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
    docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR REGISTRY\]~'${REGISTRY}'~'
./astrads/manifests/*.yaml
```

OpenShift procedure

Use the following procedure only for deployment on the Red Hat OpenShift Container Platform (OCP). Skip this procedure for deployment on non-OCP Kubernetes clusters.

- Create a namespace to deploy Astra Data Store
- Create a custom SCC
- Create roles and role bindings

Details

Create a namespace to deploy Astra Data Store

Create a namespace `astrads-system` in which all Astra Data Store components will be installed.

The following step is needed only for deployment on the Red Hat OpenShift Container Platform (OCP).

1. Create the namespace:

```
kubectl create -f ads_namespace.yaml
```

Sample: `ads_namespace.yaml`

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: operator
  name: astrads-system
```

Create a custom SCC

OpenShift uses security context constraints (SCC) that control the actions that a pod can perform. By default, the execution of any container will be granted the restricted SCC and only the capabilities defined by that SCC.

Restricted SCC does not provide permissions required by Astra Data Store cluster pods. Use this procedure to provide the required privileges (listed in the sample) to Astra Data Store.

Assign a custom SCC to the default service account for the Astra Data Store namespace.

The following steps are needed only for deployment on the Red Hat OpenShift Container Platform (OCP).

1. Create a custom SCC:

```
kubectl create -f ads_privileged_scc.yaml
```

Sample: `ads_privileged_scc.yaml`

```

allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
- '*'
allowedUnsafeSysctls:
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'ADS privileged. Grant with caution.'
    release.openshift.io/create-only: "true"
  name: ads-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups:
  type: RunAsAny
users:
- system:serviceaccount:astrads-system:default
volumes:
- '*'

```

2. Display the newly added SCC using the `oc get scc` command:

```
# oc get scc/ads-privileged
NAME          PRIV  CAPS  SELINUX  RUNASUSER  FSGROUP  SUPGROUP
PRIORITY  READONLYROOTFS  VOLUMES
ads-privileged true  ["*"] RunAsAny RunAsAny RunAsAny RunAsAny
<no value> false          ["*"]
#
```

Create the roles and role bindings

Create the required roles and role bindings to be used by the default service account for Astra Data Store.

The following yaml definition assigns various roles (via rolebindings) needed by the Astra Data Store resources in the `astrads.netapp.io` API group.

The following steps are needed only for deployment on the Red Hat OpenShift Container Platform (OCP).

1. Create the defined roles and role binding:

```
kubectl create -f oc_role_bindings.yaml
```

Sample: `oc_role_bindings.yaml`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: privcrole
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - ads-privileged
  resources:
  - securitycontextconstraints
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-scc-rolebinding
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: privcrole
```

```

subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ownerref
  namespace: astrads-system
rules:
- apiGroups:
  - astrads.netapp.io
  resources:
  - '*/finalizers'
  verbs:
  - update
---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: or-rb
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ownerref
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system

```

Configure a private image registry

As an optional step for certain environments, you can change the configuration to pull an image from a private registry that uses secrets.

1. Create the `astrads-system` namespace unless you did so already in a previous step:

```
kubectl create namespace astrads-system
```

2. Create the secret:

```
kubectl create secret docker-registry <secret-name> -n astrads-system
--docker-server=<registry name> --docker-username= <registry username>
--docker-password=<registry user password>
```

3. Add secrets configuration information to the service account:

```
kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name":
"<secret-name>"}]}' -n astrads-system
```



These changes will be applied when you [install the Astra Data Store operator](#).

Install the Astra Data Store operator

1. List the Astra Data Store manifests:

```
ls astrads/manifests/*.yaml
```

Response:

```
astrads/manifests/monitoring_operator.yaml
astrads/manifests/astradscluster.yaml
astrads/manifests/astradsversion.yaml
astrads/manifests/astradsoperator.yaml
astrads/manifests/vasa_asup_certs.yaml
astrads/manifests/manifest.yaml
astrads/manifests/configuration.yaml
```

2. Deploy the operator using kubectl apply:

```
kubectl apply -f ./astrads/manifests/astradsoperator.yaml
```

Response:



The namespace response might differ depending on whether you performed the standard installation or the [OpenShift Container Platform installation](#).

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsaddrives.astrads.n
etapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
s.netapp.io created
```



```
customresourcedefinition.apiextensions.k8s.io/astradscLOUDsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradskeyproviders.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslICENSES.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsOPTIONS.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodemANAGEMENTS.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssqOSPolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssearkeyrotaterequests.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsversions.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.astrads.netapp.io created
role.rbac.authorization.k8s.io/astrads-astrads-system-admin-role created
role.rbac.authorization.k8s.io/astrads-astrads-system-reader-role created
role.rbac.authorization.k8s.io/astrads-astrads-system-writer-role created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
role.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-admin-clusterrole created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-reader-clusterrole created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-writer-clusterrole created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-viewer-
```

```
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemangement-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemangement-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsqospolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumeeditor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumeeditor-
viewer-
```

```

role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
rolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
secret/astrads-autosupport-certs created
secret/astrads-webhook-server-cert created
service/astrads-webhook-service created
deployment.apps/astrads-operator created

```

3. Verify that the Astra Data Store operator pod has started and is running:

```
kubectl get pods -n astrads-system
```

Response:

NAME	READY	STATUS	RESTARTS	AGE
astrads-operator-5ffb94fbf-7ln4h	1/1	Running	0	17m

Deploy the Astra Data Store version YAML

1. Deploy using `kubectl apply`:

```
kubectl apply -f ./astrads/manifests/astradsversion.yaml
```

2. Verify that the pods are running:

```
kubectl get pods -n astrads-system
```

Response:

NAME	READY	STATUS	RESTARTS
AGE			
astrads-cluster-controller-7f6f884645-xxf2n 117s	1/1	Running	0
astrads-ds-nodeinfo-2jqnk 2m7s	1/1	Running	0
astrads-ds-nodeinfo-dbk7v 2m7s	1/1	Running	0
astrads-ds-nodeinfo-rn9tt 2m7s	1/1	Running	0
astrads-ds-nodeinfo-vsmhv 2m7s	1/1	Running	0
astrads-license-controller-fb8fd56bc-bxq7j 2m2s	1/1	Running	0
astrads-operator-5ffb94fbf-7ln4h 2m10s	1/1	Running	0

Apply the Astra Data Store license

1. Apply the NetApp License File (NLF) that you obtained from NetApp. Before you run the command, enter the name of the cluster (`<Astra-Data-Store-cluster-name>`) that you are [going to deploy](#) or have already deployed and the path to the license file (`<file_path/file.txt>`):

```
kubectl astrads license add --license-file-path <file_path/file.txt>  
--ads-cluster-name <Astra-Data-Store-cluster-name> -n astrads-system
```

2. Verify that the license has been added:

```
kubectl astrads license list
```

Response:

NAME	ADSCUSTER	VALID	PRODUCT
EVALUATION	ENDDATE	VALIDATED	
e100000006-ads-capacity	astrads-example-cluster	true	Astra Data Store
true	2023-01-23	2022-04-04T14:38:54Z	

Install the Astra Data Store cluster

1. Open the YAML file:

```
vim ./astrads/manifests/astradscluster.yaml
```

2. Edit the following values in the YAML file.



A simplified example of the YAML file follows these steps.

- a. (Required) **Metadata:** In `metadata`, change the `name` string to the name of your cluster. This must be the same cluster name you use when you [apply the license](#).
- b. (Required) **Spec:** Change the following required values in `spec`:
 - Change the `adsNodeConfig` values to those required for your installation, depending on your license and Astra Data Store installation size:
 - Small: 9 CPU and 38 memory
 - Medium: 23 CPU and 94 memory
 - (Optional) Remove the commenting around the `adsNodeSelector` section. Configure this if you want to constrain Astra Data Store to install only on a selected pool of worker nodes.
 - (Optional) Specify a specific number of nodes between 4-16 that should be used in the Astra Data Store cluster.
 - Change the `mvip` string to the IP address of a floating management IP that is routable from any worker node in the cluster.
 - In `adsDataNetworks`, add a comma-separated list of floating IP addresses (`addresses`) that are routable from any host where you intend to mount a NetApp volume. Use one floating IP address per node. There should be at least as many data network IP addresses as there are Astra Data Store nodes. For Astra Data Store, this means at least 4 addresses or up to 16 if you plan on expanding the cluster later.
 - In `adsDataNetworks`, specify the netmask used by the data network.
 - In `adsNetworkInterfaces`, replace the `<mgmt_interface_name>` and `<cluster_and_storage_interface_name>` values with the network interface names you want to use for management, cluster, and storage. If no names are specified, the node's primary interface will be used for management, cluster, and storage networking. Be sure to also remove the commenting around the `adsNetworkInterfaces` section.



Cluster and storage networks need to be on the same interface. The Astra Data Store management interface should be same as the Kubernetes node's management interface.

- c. (Optional) **monitoringConfig**: If you want to configure a [monitoring operator](#) (optional if you are not using Astra Control Center for monitoring), remove the commenting from the section, add the namespace in which the agent CR (monitoring operator resource) is applied (default is `netapp-monitoring`), and add the repo path for your registry (`your_registry_path`) that you used in previous steps.
- d. (Optional) **autoSupportConfig**: Retain the [AutoSupport](#) default values unless you need to configure a proxy:
 - For `proxyURL`, set the URL of the proxy with the port that will be used for AutoSupport bundle transfer.



For brevity, some comments have been removed from the YAML sample below.

```
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 38
    # [Optional] Specify node selector labels to select the nodes for
    # creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue
    adsNodeCount: 4
    mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
        # Specify the network interface names to use for management, cluster
        # and storage networks.
        # If none are specified, the node's primary interface will be used for
        # management, cluster and storage networking.
        # To move the cluster and storage networks to a different interface
        # than management, specify all three interfaces to use here.
        # NOTE: The cluster and storage networks need to be on the same
        # interface.
  adsNetworkInterfaces:
    managementInterface: "<mgmt_interface_name>"
    clusterInterface: "<cluster_and_storage_interface_name>"
```

```

storageInterface: "<cluster_and_storage_interface_name>"
# [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
# monitoringConfig:
  # namespace: "netapp-monitoring"
  # repo: "[YOUR REGISTRY]"
autoSupportConfig:
  autoUpload: true
  enabled: true
  coredumpUpload: false
  historyRetentionCount: 25
  destinationURL: "https://support.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  periodic:
    - schedule: "0 0 * * *"
      periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
            userMessage: Daily Monitoring Storage AutoSupport bundle
            nodes: all
        - component:
            name: controlplane
            event: daily
            userMessage: Daily Control Plane AutoSupport bundle

```

3. Deploy the cluster using `kubectl apply`:

```
kubectl apply -f ./astrads/manifests/astradscluster.yaml
```

4. Wait a few minutes for the cluster creation operation to complete and then verify that the pods are running:

```
kubectl get pods -n astrads-system
```

Sample response:

NAME	READY	STATUS	
RESTARTS AGE			
astrads-cluster-controller-7c67cc7f7b-2jww2 7h31m	1/1	Running	0
astrads-deployment-support-788b859c65-2qjkn 12d	3/3	Running	19
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 5d1h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-pwk42 5d2h	1/1	Running	0
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 8h	1/1	Running	0
astrads-ds-nodeinfo-gcmj8 12d	1/1	Running	1
astrads-ds-nodeinfo-j826x 12d	1/1	Running	3
astrads-ds-nodeinfo-vdthh 12d	1/1	Running	3
astrads-ds-nodeinfo-xwgsf 12d	1/1	Running	0
astrads-ds-support-828vw 5d2h	2/2	Running	2
astrads-ds-support-astrads-example-cluster-cfzts 8h	2/2	Running	0
astrads-ds-support-astrads-example-cluster-nzkkr 7h49m	2/2	Running	15
astrads-ds-support-astrads-example-cluster-xxbnp 5d2h	2/2	Running	1
astrads-license-controller-86c69f76bb-s6fb7 8h	1/1	Running	0
astrads-operator-79ff8fbb6d-vpz9m 8h	1/1	Running	0

5. Verify the cluster deployment progress:

```
kubectl get astradscluster -n astrads-system
```

Sample response:

NAME	STATUS	VERSION	SERIAL NUMBER	MVIP
AGE				
astrads-example-cluster	created	2022.05.0-X	e100000006	
10.x.x.x	13m			

Understand deployment-related events

During cluster deployment, the operation status should change from `blank` to `in progress` to `created`. Cluster deployment will last approximately 8 to 10 minutes. To monitor cluster events during deployment, you can run either of the following commands:

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n
astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

The following are key events during deployment:

Event	Message and significance
ControlPlaneNodesSelected	Successfully selected [number] control plane nodes to join the ADS cluster. The Astra Data Store operator identified enough nodes with CPU, memory, storage, and networking to create an Astra Data Store cluster.
ADSClusterCreateInProgress	The Astra Data Store cluster controller has started the cluster create operation.
ADSClusterCreateSuccess	The cluster was created successfully.

If the cluster's status doesn't change to `in progress`, check the operator logs for more details on node selection:

```
kubectl logs -n astrads-system <astrads operator pod name>
```

If the cluster's status is stuck as `in progress`, check the cluster controller's logs:

```
kubectl logs -n astrads-system <astrads cluster controller pod name>
```

Install Astra Data Store using Astra Control Center

To deploy and use Astra Data Store with Astra Control Center, do the following.

What you'll need

- You have reviewed [general Astra Data Store prerequisites](#).
- You have installed Astra Control Center.

Steps

1. [Deploy Astra Data Store using Astra Control Center](#).

What's next

- **Kubernetes-native deployments and third-party distributions:** Complete the Astra Data Store deployment by performing additional [setup tasks](#).
- **Astra Control Center:** If you have used Astra Control Center to deploy Astra Data Store, you do not need to follow these [setup tasks](#) unless you want to configure any additional monitoring options. After you deploy Astra Data Store, you can use the Astra Control Center UI to accomplish these tasks:
 - [Monitor the health of your Astra Data Store assets](#).
 - [Manage the Astra Data Store backend storage](#).
 - [Monitor nodes, disks, and persistent volume claims \(PVCs\)](#).

Set up Astra Data Store components

After you have [installed a standalone Astra Data Store](#) and addressed some [environmental prerequisites](#), you'll install Astra Trident, configure Kubernetes snapshot capabilities, set up the storage backend, and create a default storage class:



If you are using Astra Control Center to deploy Astra Data Store, you do not need to follow these steps unless you intend to set up [additional monitoring options](#).

- [Install Astra Trident](#)
- [Install Kubernetes snapshot CRDs and Controller](#)
- [Set up Astra Data Store as storage backend](#)
- [Create a default Astra Data Store storage class](#)
- [Configure Astra Data Store monitoring](#)

Install Astra Trident

For Astra Data Store, you'll need to install Astra Trident 21.10.1 or later. You can install Astra Trident using one of the following options:

- [Install Astra Trident using tridentctl](#).
- [Install Astra Trident using Trident operator](#).



You can deploy the Trident operator either manually or using Helm.

Install Kubernetes snapshot CRDs and Controller

Kubernetes snapshot CRDs and controller are required to create persistent volume claim (PVC) snapshots. If you do not already have the CRD and controller installed for your environment, run the following commands to

install them.



The following command examples assume `/trident` as the directory; however, the directory you use can be any directory that you used to download the YAML files.

What you'll need

- Before you begin installation, prepare your environment for Astra Data Store deployment.
- Download the [Kubernetes snapshot controller YAML files](#):
 - `setup-snapshot-controller.yaml`
 - `rbac-snapshot-controller.yaml`
- Download the [YAML CRDs](#):
 - `snapshot.storage.k8s.io_volumesnapshotclasses.yaml`
 - `snapshot.storage.k8s.io_volumesnapshotcontents.yaml`
 - `snapshot.storage.k8s.io_volumesnapshots.yaml`

Steps

1. Apply `snapshot.storage.k8s.io_volumesnapshotclasses.yaml`:

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
```

Response:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotclasses.snapshot.storage.k8s.io configured
```

2. Apply `snapshot.storage.k8s.io_volumesnapshotcontents.yaml`:

```
kubectl apply -f
trident/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
```

Response:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotcontents.snapshot.storage.k8s.io configured
```

3. Apply `snapshot.storage.k8s.io_volumesnapshots.yaml`:

```
kubectl apply -f trident/snapshot.storage.k8s.io_volumesnapshots.yaml
```

Response:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshots.snapshot.s  
torage.k8s.io configured
```

4. Apply setup-snapshot-controller.yaml:

```
kubectl apply -f trident/setup-snapshot-controller.yaml
```

Response:

```
deployment.apps/snapshot-controller configured
```

5. Apply rbac-snapshot-controller.yaml:

```
kubectl apply -f trident/rbac-snapshot-controller.yaml
```

Response:

```
serviceaccount/snapshot-controller configured  
clusterrole.rbac.authorization.k8s.io/snapshot-controller-runner  
configured  
clusterrolebinding.rbac.authorization.k8s.io/snapshot-controller-role  
configured  
role.rbac.authorization.k8s.io/snapshot-controller-leaderelection  
configured  
rolebinding.rbac.authorization.k8s.io/snapshot-controller-leaderelection  
configured
```

6. Verify that the CRD YAML files are applied:

```
kubectl get crd | grep volumesnapshot
```

Sample response:

```
astradsvolumesnapshots.astrads.netapp.io      2021-08-04T17:48:21Z
volumesnapshotclasses.snapshot.storage.k8s.io 2021-08-04T22:05:49Z
volumesnapshotcontents.snapshot.storage.k8s.io 2021-08-04T22:05:59Z
volumesnapshots.snapshot.storage.k8s.io       2021-08-04T22:06:17Z
```

7. Verify that the snapshot controller files are applied:

```
kubectl get pods -n kube-system | grep snapshot
```

Sample response:

```
snapshot-controller-7f58886ff4-cdh78
1/1      Running    0          13s
snapshot-controller-7f58886ff4-tmrd9
1/1      Running    0          32s
```

Set up Astra Data Store as storage backend

Configure storage backend parameters in the `ads_backend.json` file and create the Astra Data Store storage backend.

Steps

1. Create `ads_backend.json` using a secure terminal:

```
vi ads_backend.json
```

2. Configure the JSON file:



A sample JSON follows these steps.

- a. Change the `"cluster"` value to the cluster name for the Astra Data Store cluster.
- b. Change the `"namespace"` value to the namespace you want to use with volume creation.
- c. Change the `"autoExportPolicy"` value to `true` unless you set up an `exportpolicy` CR instead for this backend.
- d. Populate the `"autoExportCIDRs"` list with IP addresses you want to grant access. Use `0.0.0.0/0` to allow all.
- e. For the `"kubeconfig"` value, do the following:

- i. Convert and minimize the .kube/config YAML file to JSON format without spaces:

Example conversion:

```
python3 -c 'import sys, yaml, json;
json.dump(yaml.load(sys.stdin), sys.stdout, indent=None)' <
~/.kube/config > kubeconf.json
```

- ii. Encode as base64 and use the base64 output for the "kubeconfig" value:

Example encoding:

```
cat kubeconf.json | base64 | tr -d '\n'
```

```

{
  "version": 1,
  "storageDriverName": "astrads-nas",
  "storagePrefix": "",
  "cluster": "example-1234584",
  "namespace": "astrads-system",
  "autoExportPolicy": true,
  "autoExportCIDRs": ["0.0.0.0/0"],
  "kubeconfig": "<base64_output_of_kubeconf_json>",
  "debugTraceFlags": {"method": true, "api": true},
  "labels": {"cloud": "on-prem", "creator": "trident-dev"},
  "defaults": {
    "qosPolicy": "silver"
  },
  "storage": [
    {
      "labels": {
        "performance": "extreme"
      },
      "defaults": {
        "qosPolicy": "gold"
      }
    },
    {
      "labels": {
        "performance": "premium"
      },
      "defaults": {
        "qosPolicy": "silver"
      }
    },
    {
      "labels": {
        "performance": "standard"
      },
      "defaults": {
        "qosPolicy": "bronze"
      }
    }
  ]
}

```

3. Change to the directory where you downloaded the Trident installer:

```
cd <trident-installer or path to folder containing tridentctl>
```

4. Create the storage backend:

```
./tridentctl create backend -f ads_backend.json -n trident
```

Sample response:

```
+-----+-----+
+-----+-----+-----+-----+
|      NAME      | STORAGE DRIVER |              UUID              |
| STATE  | VOLUMES |              |
+-----+-----+
+-----+-----+-----+-----+
| example-1234584 | astrads-nas    | 2125fa7a-730e-43c8-873b-6012fcc3b527 |
| online  |          0 |              |
+-----+-----+
+-----+-----+-----+-----+
```

Create a default Astra Data Store storage class

Create the Astra Trident default storage class and apply it to the storage backend.

Steps

1. Create the trident-csi storage class:
 - a. Create ads_sc_example.yaml:

```
vi ads_sc_example.yaml
```

Example:


```

allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2022-05-09T18:05:21Z"
  name: ads-silver
  resourceVersion: "3361772"
  uid: 1o023456-da4b-51e3-b430-3aa1e3bg111a
mountOptions:
- vers=4
parameters:
  backendType: astrads-nas
  selector: performance=premium
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate

```

b. Create trident-csi:

```
kubectl create -f ads_sc_example.yaml
```

Response:

```
storageclass.storage.k8s.io/trident-csi created
```

2. Verify that the storage class has been added:

```
kubectl get storageclass
```

Response:

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
ads-silver	csi.trident.netapp.io	Delete	Immediate
allowVolumeExpansion	AGE		
true	6h29m		

3. Change to the directory where you downloaded the Trident installer:

```
cd <trident-installer or path to folder containing tridentctl>
```

4. Verify that the Astra Trident backend has been updated with the default storage class parameters:

```
./tridentctl get backend -n trident -o yaml
```

Sample response:

```
items:
- backendUUID: 2125fa7a-730e-43c8-873b-6012fcc3b527
  config:
    autoExportCIDRs:
    - 0.0.0.0/0
    autoExportPolicy: true
    backendName: ""
    cluster: example-1234584
    credentials: null
    debug: false
    debugTraceFlags:
      api: true
      method: true
    defaults:
      exportPolicy: default
      qosPolicy: bronze
      size: 1G
      snapshotDir: "false"
      snapshotPolicy: none
    disableDelete: false
    kubeconfig: <ID>
    labels:
      cloud: on-prem
      creator: trident-dev
    limitVolumeSize: ""
    namespace: astrads-system
    nfsMountOptions: ""
    region: ""
    serialNumbers: null
    storage:
    - defaults:
        exportPolicy: ""
        qosPolicy: gold
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
        region: ""
```

```

supportedTopologies: null
zone: ""
- defaults:
  exportPolicy: ""
  qosPolicy: silver
  size: ""
  snapshotDir: ""
  snapshotPolicy: ""
labels:
  performance: premium
region: ""
supportedTopologies: null
zone: ""
- defaults:
  exportPolicy: ""
  qosPolicy: bronze
  size: ""
  snapshotDir: ""
  snapshotPolicy: ""
labels:
  performance: standard
region: ""
supportedTopologies: null
zone: ""
storageDriverName: astrads-nas
storagePrefix: ""
supportedTopologies: null
version: 1
zone: ""
configRef: ""
name: example-1234584
online: true
protocol: file
state: online
storage:
  example-1234584_pool_0:
    name: example-1234584_pool_0
    storageAttributes:
      backendType:
        offer:
          - astrads-nas
      clones:
        offer: true
      encryption:
        offer: false
    labels:

```

```

    offer:
      cloud: on-prem
      creator: trident-dev
      performance: extreme
  snapshots:
    offer: true
storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_1:
  name: example-1234584_pool_1
  storageAttributes:
    backendType:
      offer:
        - astrads-nas
    clones:
      offer: true
    encryption:
      offer: false
    labels:
      offer:
        cloud: on-prem
        creator: trident-dev
        performance: premium
    snapshots:
      offer: true
storageClasses:
- trident-csi
supportedTopologies: null
example-1234584_pool_2:
  name: example-1234584_pool_2
  storageAttributes:
    backendType:
      offer:
        - astrads-nas
    clones:
      offer: true
    encryption:
      offer: false
    labels:
      offer:
        cloud: on-prem
        creator: trident-dev
        performance: standard
    snapshots:
      offer: true

```

```
storageClasses:
  - ads-silver
supportedTopologies: null
volumes: []
```

Configure Astra Data Store monitoring

(Optional) You can configure Astra Data Store for monitoring by another telemetry service. This procedure is recommended if you are not using Astra Control Center for Astra Data Store monitoring or if you want to extend monitoring to additional endpoints.

You can install the monitoring operator if your Astra Data Store instance is a standalone deployment, uses Cloud Insights to monitor telemetry, or streams logs to a third-party endpoint such as Elastic.



For Astra Control Center deployments, the monitoring operator is automatically configured. You can skip the first two commands of the following procedure.

What you'll need

Before setting up monitoring, you will need an active Astra data store cluster in the `astrads-system` namespace.

Steps

1. Run this install command:

```
kubectl apply -f ./manifests/monitoring_operator.yaml
```

2. Configure Astra Data Store for monitoring:

```
kubectl astrads monitoring -n netapp-monitoring -r [YOUR REGISTRY] setup
```

3. Configure Astra Data Store to stream EMS logs to an Elastic endpoint:

```
kubectl astrads monitoring es --port <portname> --host <hostname>
```

Astra Data Store Early Access Program (EAP) release limits

Astra Data Store has the following resource limits during the Early Access Program.

Resource	Minimum	Maximum
Number of nodes in Astra Data Store cluster	4	16
Number of persistent volumes per node	N/A	50

Resource	Minimum	Maximum
Volume size	20MiB	2TiB
Snapshots per volume	0	1023
Clones per Volume	0	9
VMs per node	0	50

Frequently asked questions for Astra Data Store

Find answers to the frequently asked questions about installing, configuring, upgrading, and troubleshooting Astra Data Store Early Access Program release.

General questions

Can I use the Astra Data Store Early Access Program release for production?

No. While Astra Data Store is designed and developed to deliver enterprise grade resiliency, the Early Access Program release of Astra Data Store is not targeted for production workloads.

Can I use Astra Data Store for virtual machine workloads?

Yes. Astra Data Store supports both Kubernetes and VMware vVol workloads.

See [Learn about Astra Data Store with VMware](#).

Can I use VMware vSphere to manage Astra Data Store?

Yes, Astra Data Store can be natively managed in vCenter with the NetApp Astra Plugin for VMware vSphere. See [Manage Astra Data Store components of your VMware installation](#).

Does Astra Data Store have any dependencies on other NetApp products for its functioning?

Yes. Astra Data Store requires NetApp CSI driver Astra Trident version 21.10.1 and later to be deployed on workload Kubernetes clusters. Learn about [Astra Data Store requirements](#).

Astra Control Center is required to enable the VMware workflows and NetApp Astra Plugin for VMware vSphere.

Applications using an Astra Data Store cluster as a storage backend can use [Astra Control Center](#) to leverage application-aware data management capabilities, including data protection, disaster recovery, and migration of Kubernetes workloads.

How can I manage the Astra Data Store cluster?

You can manage Astra Data Store resources by using `kubectl` commands and by using the Kubernetes API extension.

The `kubectl astrads` command includes an `-h` switch that provides usage and flag documentation for your convenience.

How can I monitor Astra Data Store cluster metrics?

You can monitor Astra Data Store metrics using Cloud Insights (see [Monitor metrics with Cloud Insights](#)) or Kubernetes native monitoring (see [Monitor with Prometheus and Grafana](#)).

You can also monitor logs. See [Configure and monitor event logs](#).

Can I use Astra Data Store along with ONTAP or other storage providers in my Kubernetes clusters?

Yes. Astra Data Store can be used along with other storage providers in your application clusters.

Will Astra Trident be uninstalled if I remove a Kubernetes cluster from Astra Data Store?

Astra Trident will not be uninstalled from the cluster if you uninstall Astra Data Store. If you require Astra Trident to be uninstalled, you will have to do it separately.

Licensing

Does Astra Data Store require a license?

Yes, Astra Data Store requires an evaluation NetApp license file (NLF) for the Early Access Program.

See [Astra Data Store requirements](#).

How long is the Astra Data Store evaluation license valid?

The Astra Data Store license has a default term of 90 days from the download date.

Installation and use of Astra Data Store on a Kubernetes cluster

Can I install Astra Data Store on a Kubernetes cluster running on bare metal or virtual machines?

Yes. Astra Data Store can be installed on a Kubernetes cluster running on bare metal or on vSphere VMs. See [Astra Data Store requirements](#).

What are the supported versions of Kubernetes for Astra Data Store?

Astra Data Store works with Kubernetes distributions that are compatible with v1.20 and later. However, it is not currently validated with all Kubernetes distributions. Learn about [Astra Data Store requirements](#).

My Kubernetes cluster is larger than 4 worker nodes. Can I install Astra Data Store on it?

Yes. An Astra Data Store cluster must be initially deployed on 4 worker nodes in a Kubernetes cluster. After deployment, you can grow the cluster up to a maximum of 16 worker nodes.

Does Astra Data Store support an offline installation from a private registry?

Yes. Astra Data Store can be installed offline from a local registry. See [Install Astra Data Store](#).

Do I need internet connectivity to use Astra Data Store?

No, Astra Data Store Early Access Program does not require internet connectivity. However, it is recommended to have connectivity to the NetApp AutoSupport backend to periodically send the telemetry bundles. This connectivity can be direct or via a proxy.

What are the roles and privileges used by Astra Data Store?

You need to be a kube admin to deploy the Astra Data Store operator.

Astra Data Store has a privileged daemonset called `astrads-ds-nodeinfo` for discovering node resources used in selecting nodes.

In addition, the operator will use privileged Kubernetes jobs to install the storage cluster's containers on the selected worker nodes to build the Astra Data Store storage cluster.

What manifest files do I need to update for Astra Data Store installation?

From the Astra Data Store bundle you download from the [NetApp Support Site](#), you will get the following manifests:

- `astradscluster.yaml`

- astradsoperator.yaml
- astradsversion.yaml
- monitoring_operator.yaml

You will need to update the `astradscluster.yaml` manifest with your deployment-specific configuration. See [Install Astra Data Store](#).

Troubleshooting and support

With Astra Data Store, you can access community support using the NetApp Containers Slack channel. This channel is monitored by NetApp Support and our Technical Marketing Engineers.

[NetApp Container Slack channel](#)

See [Astra Data Store support operations](#).

How do I raise a support case OR ask clarification on a quick question?

To raise a support case or get clarification on a quick question, report your issue or question on the [NetApp Container Slack channel](#). NetApp Support will follow up with you to assist on a best-effort basis.

How do I raise a request for a new feature?

If you have questions on supported configurations or capabilities, reach out to astra.feedback@netapp.com.

How do I generate a support log bundle?

See [Generate support bundle](#) for instructions on setting up and downloading support log bundles for Astra Data Store.

Astra Data Store cannot find my Kubernetes node. How do I fix this?

See [Install Astra Data Store](#).

Can IPv6 addresses be used for management, data, and cluster networks?

No, Astra Data Store supports only IPv4 address. IPv6 support will be added in a future release of Astra Data Store.

What NFS version is used while provisioning a volume on Astra Data Store?

Astra Data Store supports NFS v4.1 for all volumes provisioned for Kubernetes applications and NFSv3 for all volumes provisioned for VMware workloads.

See [Astra Data Store requirements](#) and [Astra Data Store limits](#).

Upgrading Astra Data Store

Can I upgrade from Astra Data Store preview release?

Yes. You can upgrade from the Astra Data Store Early Access Program release to a future release.

Copyright Information

Copyright © 2022 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.