



Astra Data Store documentation

Astra Data Store

NetApp
June 27, 2022

This PDF was generated from <https://docs.netapp.com/us-en/astra-data-store/index.html> on June 27, 2022. Always check docs.netapp.com for the latest.

Table of Contents

| | |
|---|-----|
| Astra Data Store documentation | 1 |
| Release notes | 2 |
| What's new in this release of Astra Data Store | 2 |
| Known issues | 3 |
| Known limitations | 4 |
| Concepts | 5 |
| Learn about Astra Data Store | 5 |
| Astra Data Store deployment models | 6 |
| Cluster expansion | 7 |
| Storage efficiency in Astra Data Store | 8 |
| Security in Astra Data Store | 8 |
| Get started | 10 |
| Astra Data Store requirements | 10 |
| Quick start for Astra Data Store | 14 |
| Install Astra Data Store | 15 |
| Set up Astra Data Store components | 33 |
| Astra Data Store Early Access Program (EAP) release limits | 44 |
| Frequently asked questions for Astra Data Store | 45 |
| Use Astra Data Store | 48 |
| Manage Astra Data Store resources with kubectl commands | 48 |
| Deploy a test application | 54 |
| Manage the Astra Data Store cluster | 58 |
| Monitor Astra Data Store | 70 |
| Secure Astra Data Store | 83 |
| Update the Astra Data Store license | 88 |
| Upgrade Astra Data Store | 89 |
| Uninstall Astra Data Store with an automated script | 91 |
| Use Astra Data Store with VMware | 107 |
| Learn about Astra Data Store with VMware | 107 |
| Astra Data Store with VMware requirements | 107 |
| Set up Astra Data Store with VMware | 108 |
| Monitor components of your VMware installation | 115 |
| Manage Astra Data Store components of your VMware installation | 117 |
| Uninstall Astra Data Store from a VMware-integrated environment | 120 |
| Knowledge and support | 121 |
| Troubleshooting | 121 |
| Get help | 121 |
| Automatic support monitoring | 121 |
| Earlier versions of Astra Data Store | 126 |
| Legal notices | 127 |
| Copyright | 127 |
| Trademarks | 127 |
| Patents | 127 |

| | |
|----------------------|-----|
| Privacy policy | 127 |
| Open source | 127 |

Astra Data Store documentation

Release notes

We're pleased to announce the 2022.05.01 Early Access Program (EAP) release of Astra Data Store.

- [What's new in this release of Astra Data Store](#)
- [Known issues](#)
- [Known limitations](#)

Follow us on Twitter @NetAppDoc. Send feedback about documentation by becoming a [GitHub contributor](#) or sending an email to doccomments@netapp.com.

What's new in this release of Astra Data Store

We're pleased to announce the 2022.05.01 Early Access Program (EAP) release of Astra Data Store.

28 June 2022 (2022.05.01 EAP Release)

This patch release (2022.05.01) for the Astra Data Store EAP release (22.05) includes the following improvements:

- Astra Data Store and Astra Control Center can now be run in the same cluster.
- Delete VVol datastore workflow from vSphere is now available.



This release requires an updated version of the Astra Control Center that is packaged with the kubectl-astra plugin.

31 May 2022 (2022.05 EAP Release)

The Astra Data Store 2022.05 release includes these features:

- VMware environment integrations:
 - NFS vVol datastore workflows and Storage Policy Based Management storage provisioning.
 - Astra Plugin for VMware vSphere for native storage management and monitoring from vCenter.
- Enhanced cluster management, including [GUI-based cluster deployment from Astra Control Center](#).
- Support for [larger cluster scale](#) (higher number of nodes, increased CPU and capacity).
- Security enhancements (keys and certificates and support for external key management (KMIP)).

08 February 2022 (2021.12.1)

Patch release (2021.12.1) for the Astra Data Store preview (21.12).

- With this release, Astra Data Store preview supports VXLAN configurations with Calico CNI.
- Calico WireGuard enabled network configurations are now supported by Astra Data Store preview.
- Updated AstraDSCluster.yaml `adsNetworkInterfaces` section includes improved, descriptive comments.
- New Astra Data Store uninstall script now provided as an easier alternative to the kubectl command uninstall process.

21 December 2021 (21.12)

Initial release of Astra Data Store preview.

- [What it is](#)
- [Deployment models and components](#)
- [What it takes to get started](#)
- [Install and setup](#)
- [Manage](#) and [monitor](#) performance
- [Monitor with Cloud Insights](#)
- [Get help](#) and [use automatic support monitoring](#)

Find more information

- [Known issues](#)
- [Known limitations](#)
- [NetApp Knowledge Base articles](#)

Known issues

Known issues identify problems that might prevent you from using this release of the product successfully.

Astra Data Store 2202.05 Early Access Program (EAP) release

This release includes these known issues. Review these known issues carefully.

Storage backend tasks not showing in vSphere

When an action to use an existing storage backend or remove a storage backend is submitted, the task does not appear in the Recent Tasks panel of vSphere client.

Extra folders are created when creating a VM

When you create a VM in vSphere, additional folders are sometimes created instead of a single folder per VM under the datastore.

Filtered views in Astra Plugin for VMware vSphere intermittently do not show data

The table column filters for virtual machines occasionally omit filtered data. The filtered view shows the loading icon instead. To resolve this, you can go to another view and reset filters to show the data again.

IP address restrictions on Kubernetes worker nodes

Astra Data Store cluster creation fails in the following scenario:

1. Kubernetes worker node has an IP address hosted on an interface not assigned to Astra Data Store.
2. The IP address is in the same subnet as the cluster/data IP addresses assigned to the Astra Data Store cluster.

Find more information

- [Known limitations](#)

Known limitations

Known limitations identify platforms, devices, or functions that are not supported by this Early Access Program (EAP) release of the product or do not interoperate correctly with it.

Astra Data Store 2022.05.01 Early Access Program (EAP) release

This release includes these known limitations. Review these known limitations carefully.

VMware integrations limitations

The following features are not supported:

- VM workflows:
 - Clone a VM, including clones from a template
 - Attaching a VM disk after previously removing it
- vVol replication
- First Class Disks (FCDs)
- VMware vMotion
- Linked Mode vCenter servers
- Compliance alerts and notifications
- Multi-vCenter and multi-VASA Provider configurations
 - A single VASA Provider per vCenter configuration is supported
- Storage policy-based management is applicable only at the VM level and cannot be modified after VM creation.

Cannot resize multiple VM disks simultaneously

For this release, you can only resize one disk at a time. If you try to resize multiple disks at once, the resize operation works only for the first disk.

Find more information

- [Known issues](#)

Concepts

Learn about Astra Data Store

Astra Data Store is Kubernetes-native, shared file software-defined storage (SDS) solution for on-premises data centers that helps customers manage their cloud-native applications. Astra Data Store provides a native shared file service for both container and VM workloads along with NetApp enterprise data management.

With Astra Data Store, you can do the following:

- **Support Kubernetes containerized workloads:** With enterprise data management services and tools you are used to.
- **Use Kubernetes "application-as-a-service" platform for DevOps:** Create elastic, software-defined, self-serve platforms that deliver automated, repeatable services, removing complexity from developers.

Astra Data Store is part of the Astra product family. Learn about the [Astra family](#).

Astra Data Store features

Astra Data Store provides end-to-end Kubernetes-native storage and data management for your cloud-native applications with these features:

- **Kubernetes-native shared file service:** Provides a shared file service native to Kubernetes using a standard NFS client as a unified data store for containers and VMs.
- **Cloud scale:** Offers Kubernetes-native multiple parallel file systems on the same resource pool to achieve cloud-like scale and utilization, removing the need to manage storage separately from the cluster.
- **API-first approach:** Delivers infrastructure as code for automated cluster and workload management.
- **Enterprise-grade data management:** Delivers automated application-aware data protection and disaster recovery:
 - **NetApp technologies:** Leverages NetApp data management technology for snapshots, backups, replication, and cloning so users can build and deploy enterprise apps in Kubernetes.
 - **Resiliency:** Uses replication and erasure coding technologies for Kubernetes-native workloads for greater resiliency.
 - **Data efficiency:** Controls cost as you scale through inline deduplication and compression features.
- **Fits into your existing environment:** Supports your microservices-based and traditional workloads, serves major Kubernetes distributions, provides file storage, and runs on your choice of hardware.
- **Integration with NetApp Cloud Insights:** Delivers observability, analytics and monitoring for continuous optimization.

Get started with Astra Data Store

First, [learn about Astra Data Store requirements](#).

Then, [get started](#).

For more information

- [Astra family introduction](#)
- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [Cloud Insights documentation](#)
- [ONTAP documentation](#)

Astra Data Store deployment models

Astra Data Store manages storage drives directly on the hosts using an application deployed and orchestrated with Kubernetes.

You can install Astra Data Store on bare metal or virtual servers by using one of the following options:

- Deploy on a standalone dedicated Kubernetes cluster serving persistent volumes to Kubernetes applications running in a separate cluster (standalone cluster).
- Deploy on a Kubernetes cluster also hosting other workload applications on the same node pool (converged cluster).
- Deploy on a Kubernetes cluster also hosting other workload applications on a different node pool (disaggregated cluster).

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

Astra Data Store is part of the Astra product family. For a perspective on the entire Astra family, see [Astra family introduction](#).

Astra Data Store ecosystem

Astra Data Store works with the following:

- **Astra Control Center:** Use Astra Control Center software for application-aware data management of Kubernetes clusters in your on-premise environment. Easily back up Kubernetes apps, migrate data to a different cluster, and instantly create working application clones.

Astra Control Center supports Kubernetes clusters with an Astra Trident storage backend with ONTAP or an Astra Data Store storage backend.

- **Astra Trident:** As a fully supported open source storage provisioner and orchestrator maintained by NetApp, Astra Trident enables you to create storage volumes for containerized applications managed by Docker and Kubernetes.

You use Astra Trident to create volumes on Astra Data Store.

- **Cloud Insights:** A NetApp cloud infrastructure monitoring tool, Cloud Insights enables you to monitor performance and utilization for your Kubernetes clusters managed by Astra Control. Cloud Insights correlates storage usage to workloads.

When you enable the Cloud Insights connection in Astra Control, telemetry information shows in Astra

Control UI pages. Cloud Insights displays information about resources managed in Astra Data Store.

Astra Data Store interfaces

You can complete tasks using different interfaces:

- **Web user interface (UI):** Astra Control Service and Astra Control Center use the same web-based UI where you can manage, migrate and protect apps. This UI also displays information about Astra Data Store volumes.
- **API:** Astra Control Service and Astra Control Center use the same Astra Control API. Using the API, you can perform the same tasks that you would using the UI. You can also retrieve information about Astra Data Store with the Astra Control API.
- **kubectl commands:** To work with Astra Data Store, you can use kubectl commands directly.
- **Kubernetes extension:** Additionally, you can use the Astra Data Store kubernetes API extension.

Custom Resource Definitions (CRDs) are extensions to the kubernetes REST API that are created when the Astra Data Store operator is deployed. External entities interact with the CRDs by making calls to the Kubernetes API server. Astra Data Store watches for updates to specific CRDs and then makes calls to internal REST APIs.

For more information

- [Astra family introduction](#)
- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [Cloud Insights documentation](#)
- [ONTAP documentation](#)

Cluster expansion

Astra Data Store supports nodes of different types and capabilities in a cluster. If you are expanding a cluster, Astra Data Store supports adding nodes with any performance capabilities, as long as they are not lower than the least performant node in the cluster. New nodes must have the same storage capacity as existing nodes. All nodes, including new nodes during an expansion, need to at least meet the minimum requirements in [Astra Data Store requirements](#).

For more information

- [Astra family introduction](#)
- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra API](#)
- [Cloud Insights documentation](#)

- [ONTAP documentation](#)

Storage efficiency in Astra Data Store

Astra Data Store uses storage efficiency technologies based on NetApp ONTAP and SolidFire technologies including:

- **Thin provisioning:** A thin-provisioned volume is one for which storage is not reserved in advance. Instead, storage is allocated dynamically, as it is needed. Free space is released back to the storage system when data in the volume or LUN is deleted.
- **Zero-block detection and elimination:** Astra Data Store storage systems with thin provisioning provide the ability to detect areas of a volume that have been zeroed out so they can reclaim that space and use it elsewhere.
- **Compression:** Compression reduces the amount of physical storage required for a volume by combining data blocks in compression groups, each of which is stored as a single block. Reads of compressed data are faster than in traditional compression methods because Astra Data Store decompresses only the compression groups that contain the requested data, not an entire file.
- **Deduplication:** Deduplication reduces the amount of storage required for a volume (or all the volumes in an AFF aggregate) by discarding duplicate blocks and replacing them with references to a single shared block. Reads of deduplicated data typically incur no performance charge. Writes incur a negligible charge except on overloaded nodes.

All of these features are enabled by default.

See [storage efficiency details](#).

For more information

- [Astra family introduction](#)
- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [ONTAP documentation](#)

Security in Astra Data Store

Astra Data Store uses several methods to secure client and administrator access to storage, protect communication and data, and protect against viruses.

Astra Data Store uses the following security methods:

- Communication encryption using mutual transport layer security (mTLS)
- Role-based access control, which governs access to features
- Deployment security
- Certificate management
- Software encryption at rest including internal and external key management

For more information

- [Astra family introduction](#)
- [Astra Control Service documentation](#)
- [Astra Control Center documentation](#)
- [Astra Trident documentation](#)
- [Use the Astra Control API](#)
- [Cloud Insights documentation](#)
- [ONTAP documentation](#)

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.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradskeyproviders.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslICENSES.astrads.ne
tapp.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.ast
rads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsSQOSPolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssearkeyrotatereques
ts.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsversions.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumeFILES.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.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.

Use Astra Data Store

Manage Astra Data Store resources with kubectl commands

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

To learn how to deploy a sample app, see [Deploy a test application](#).

For cluster maintenance information, see [Manage the cluster](#).

What you'll need

- The Astra Data Store kubectl plugin you installed in [Install Astra Data Store](#)

List Kubernetes custom API resources for Astra Data Store

You can use kubectl commands inside of Kubernetes to interact with and observe the state of your Astra Data Store cluster.

Each item listed from the `api-resources` command represents a Kubernetes custom resource definition (CRD) that Astra Data Store uses internally to manage your cluster.

This list is particularly helpful to get short names of each Astra Data Store object to reduce your typing, as shown later.

1. Display a list of Kubernetes custom API resources for Astra Data Store:

```
kubectl api-resources --api-group astrads.netapp.io
```

Response:

| NAME | SHORTNAMES | APIGROUP | NAMESPACED | KIND |
|------------------------|------------|----------------------------|------------|------|
| astradsautosupports | adsas | astrads.netapp.io/v1alpha1 | true | |
| AstraDSAutoSupport | | | | |
| astradscLOUDsnapshots | adscs | astrads.netapp.io/v1alpha1 | true | |
| AstraDSCloudSnapshot | | | | |
| astradsclusters | adscl | astrads.netapp.io/v1alpha1 | true | |
| AstraDSCluster | | | | |
| astradsexportpolicies | adsep | astrads.netapp.io/v1alpha1 | true | |
| AstraDSEExportPolicy | | | | |
| astradsfaileddrives | adsfd | astrads.netapp.io/v1alpha1 | true | |
| AstraDSFailedDrive | | | | |
| astradsllicenses | adsli | astrads.netapp.io/v1alpha1 | true | |
| AstraDSLICENSE | | | | |
| astradsnfsoptions | adsnf | astrads.netapp.io/v1alpha1 | true | |
| AstraDSNfsOption | | | | |
| astradsnodeinfoes | adsni | astrads.netapp.io/v1alpha1 | true | |
| AstraDSNodeInfo | | | | |
| astradsnodemanagements | adsnm | astrads.netapp.io/v1alpha1 | true | |
| AstraDSNodeManagement | | | | |
| astradsqospolicies | adsqp | astrads.netapp.io/v1alpha1 | true | |
| AstraDSQosPolicy | | | | |
| astradsversions | adsve | astrads.netapp.io/v1alpha1 | true | |
| AstraDSVersion | | | | |
| astradsvolumeFiles | adsvf | astrads.netapp.io/v1alpha1 | true | |
| AstraDSVolumeFiles | | | | |
| astradsvolumes | adsvo | astrads.netapp.io/v1alpha1 | true | |
| AstraDSVolume | | | | |
| astradsvolumesnapshots | adsvs | astrads.netapp.io/v1alpha1 | true | |
| AstraDSVolumeSnapshot | | | | |

2. To get all the current Astra Data Store objects in your Kubernetes cluster, use the `kubectl get ads -A` command:

```
kubectl get ads -A
```

Response:

| NAMESPACE | NAME | AGE |
|----------------|---|-----|
| astrads-system | astradsqospolicy.astrads.netapp.io/bronze | 45h |
| astrads-system | astradsqospolicy.astrads.netapp.io/gold | 45h |
| astrads-system | astradsqospolicy.astrads.netapp.io/silver | 45h |

| NAMESPACE | NAME | STATUS | VERSION | SERIAL NUMBER | MVIP | AGE |
|-----------|------|--------|---------|---------------|------|-----|
|-----------|------|--------|---------|---------------|------|-----|

```
astrads-system    astradscluster.astrads.netapp.io/astrads-cluster-9f1
created    arda-9.11.1    e000000009    10.224.8.146    46h
```

```
NAMESPACE      NAME
AGE
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
astrads-system  astradsnodeinfo.astrads.netapp.io/englab.netapp.com
46h
```

```
NAMESPACE      NAME      AGE
astrads-system  astradsversion.astrads.netapp.io/astradsversion  46h
```

```
NAMESPACE      NAME      AGE
astrads-system  astradsvolumefiles.astrads.netapp.io/test23      27h
astrads-system  astradsvolumefiles.astrads.netapp.io/test234      27h
astrads-system  astradsvolumefiles.astrads.netapp.io/test2345      4h22m
```

```
NAMESPACE      NAME      SIZE    IP
CLUSTER      CREATED
astrads-system  astradsvolume.astrads.netapp.io/test234    21Gi
172.25.123.123  astrads-cluster-9f1    true
astrads-system  astradsvolume.astrads.netapp.io/test2345    21Gi
172.25.123.123  astrads-cluster-9f1    true
```

```
NAMESPACE      NAME
SEQUENCE COMPONENT      EVENT      TRIGGER      PRIORITY      SIZE
STATE
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
adsclustercreatesuccess-20211214t 9      controlplane
adsclustercreatesuccess k8sEvent    notice    0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
daily-20211215t0      15      controlplane    daily
periodic    notice    0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/controlplane-
daily-20211216t0      20      controlplane    daily
periodic    notice    0      uploaded
astrads-system  astradsautosupport.astrads.netapp.io/storage-
callhome.dbs.cluster.cannot.sync.blocks 10      storage
callhome.dbs.cluster.cannot.sync.blocks  firetapEvent    emergency    0
uploaded
```

| NAMESPACE | NAME | ADSCUSTER |
|----------------|-------------------------------------|---------------------------------|
| VALID PRODUCT | EVALUATION ENDDATE | VALIDATED |
| astrads-system | astradslicense.astrads.netapp.io/e0 | astrads-cluster- |
| 9f1 true | Astra Data Store true | 2022-02-07 2021-12-16T20:43:23Z |

3. Use one of the short names to show the current state of volumes in the cluster:

```
kubectl get adsvo -A
```

Response:

| NAMESPACE | NAME | SIZE | IP | CLUSTER |
|----------------|----------|------|----------------|------------------|
| CREATED | | | | |
| astrads-system | test234 | 21Gi | 172.25.138.109 | astrads-cluster- |
| 9f1c99f true | | | | |
| astrads-system | test2345 | 21Gi | 172.25.138.111 | astrads-cluster- |
| 9f1c99f true | | | | |

Use the help option on the kubectl extension

NetApp provides an `astrads` extension to the `kubectl` command that enables you to perform Astra Data Store cluster management tasks such as adding a license, managing nodes, troubleshooting issues and expanding the Astra Data Store cluster. The `astrads` extension includes an `-h` option that provides information about how to use the extension and for what tasks.

1. Show help for all commands in the Astra Data Store `kubectl` extension:

```
kubectl astrads -h
```

Response:

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

Available Commands:
  asup      Manage AutoSupport
  clusters  Manage clusters
  drives    Manage drives in a cluster
  faileddrive Manage drive replacement in a cluster
  help      Help about any command
  license   Manage license in the astrads cluster
```

| | |
|-------------|-------------------------------------|
| maintenance | Manage maintenance status of a node |
| monitoring | Manage Monitoring Output |
| nodes | Manage nodes in a cluster |

Flags:

| | |
|--|---|
| <code>--as</code> string operation | Username to impersonate for the operation |
| <code>--as-group</code> stringArray operation, this flag can be groups. | Group to impersonate for the operation, this flag can be repeated to specify multiple groups. |
| <code>--cache-dir</code> string cache") | Default HTTP cache directory (default "/u/arda/.kube/http-cache") |
| <code>--certificate-authority</code> string certificate authority | Path to a cert file for the certificate authority |
| <code>--client-certificate</code> string for TLS | Path to a client certificate file for TLS |
| <code>--client-key</code> string | Path to a client key file for TLS |
| <code>--cluster</code> string cluster to use | The name of the kubeconfig cluster to use |
| <code>--context</code> string context to use | The name of the kubeconfig context to use |
| <code>-h, --help</code> | help for astrads |
| <code>--insecure-skip-tls-verify</code> certificate will not be checked your HTTPS connections insecure | If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure |
| <code>--kubeconfig</code> string use for CLI requests. | Path to the kubeconfig file to use for CLI requests. |
| <code>-n, --namespace</code> string for this CLI request | If present, the namespace scope for this CLI request |
| <code>--request-timeout</code> string before giving up on a single should contain a 1s, 2m, 3h). timeout requests. | The length of time to wait for a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests. (default "0") |
| <code>-s, --server</code> string Kubernetes API server | The address and port of the Kubernetes API server |
| <code>--token</code> string to the API server | Bearer token for authentication to the API server |
| <code>--user</code> string | The name of the kubeconfig user |

to use

2. Use `astrads [command] --help` for more information about a command.

```
kubectl astrads asup collect --help
```

Response:

Collect the autosupport bundle by specifying the component to collect. It will default to manual event.

Usage:

```
astrads asup collect [flags]
```

Examples:

```
# Control plane collection
```

```
kubectl astrads collect --component controlplane example1
```

```
# Storage collection for single node
```

```
kubectl astrads collect --component storage --nodes node1 example2
```

```
# Storage collection for all nodes
```

```
kubectl astrads collect --component storage --nodes all example3
```

```
# Collect but don't upload to support
```

```
kubectl astrads collect --component controlplane --local example4
```

NOTE:

```
--component storage and --nodes <name> are mutually inclusive.
```

```
--component controlplane and --nodes <name> are mutually exclusive.
```

Flags:

```
-c, --component string      Specify the component to collect:  
[storage , controlplane , vasaprovider, all]
```

```
-d, --duration int          Duration is the duration in hours from  
the startTime for collection  
of AutoSupport.
```

```
-e, --event string          Specify the callhome event to trigger.  
(default "manual")
```

```
-f, --forceUpload           Configure an AutoSupport to upload if  
it is in the compressed state
```

```
and not
```

```
uploading because it was created with
```



```

the 'local' option or if
disabled
    -h, --help            help for collect
    -l, --local            Only collect and compress the
autosupport bundle. Do not upload
to support.
bundle after it is in
    Use 'download' to copy the collected
the 'compressed' state
    Specify nodes to collect for storage
--nodes string            StartTime is the starting time for
component. (default "all")
    -t, --startTime string This should be in the ISO 8601 date
collection of AutoSupport.
time format.
    Example format accepted:
    2021-01-01T15:20:25Z, 2021-01-
01T15:20:25-05:00
    -u, --usermessage string UserMessage is the additional message
to include in the
AutoSupport subject.
CLI")                    (default "Manual event trigger from

```

Deploy a test application

Here are steps to deploy a test application that you can use with Astra Data Store.

In this example, we use a Helm repository to deploy a MongoDB chart from Bitnami.

What you'll need

- Astra Data Store cluster deployed and configured
- Trident installation completed

Steps

1. Add a Helm repo from Bitnami:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
```

2. Deploy MongoDB:

```
helm install mongohelm4 --set persistence.storageClass=trident-csi
bitnami/mongodb --namespace=ns-mongodb --create-namespace
```

3. Check the status of the MongoDB pod:

```
~% kubectl get pods -n ns-mongodb
```

| NAME | READY | STATUS | RESTARTS | AGE |
|-------------------------|-------|---------|----------|-----|
| mongodb-9846ff8b7-rfr4r | 1/1 | Running | 0 | 67s |

4. Verify the persistent volume claim (PVC) used by MongoDB:

```
~% kubectl get pvc -n ns-mongodb
```

| NAME | STATUS | VOLUME | CAPACITY | ACCESS MODES |
|------------------|--------|------------------------|----------|--------------|
| STORAGECLASS AGE | | | | |
| mongodb | Bound | pvc-1133453a-e2f5-48a5 | 8Gi | RWO |
| trident-csi | 97s | | | |

5. List the volume by using the kubectl command `get astradsvolume`:

```
~% kubectl get astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-system
```

| NAME | SIZE | IP | CLUSTER | CREATED |
|------------------------|-----------|--------------|---------|---------|
| pvc-1133453a-e2f5-48a5 | 8830116Ki | 10.192.2.192 | jai-ads | true |

6. Describe the volume by using the kubectl command `describe astradsvolume`:

```
~% kubectl describe astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-system
```

Name: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07

Namespace: astrads-system

Labels: astrads.netapp.io/cluster=jai-ads
astrads.netapp.io/mip=10.192.1.39
astrads.netapp.io/volumeUUID=cf33fd38-a451-596c-b656-61b8270d2b5e
trident.netapp.io/cloud=on-prem
trident.netapp.io/creator=trident-dev
trident.netapp.io/performance=premium

Annotations: provisioning: {"provisioning":{"cloud":"on-prem","creator":"trident-dev","performance":"premium"}}
trident: {"trident":{"version":"21.10.0-test.jenkins-trident-stable-v21.10-2+e03219ce37294d9ba54ec476bbe788c1a7772548","backendUUID":"","platform":

```

...
API Version:  astrads.netapp.io/v1alpha1
Kind:          AstraDSVolume
Metadata:
  Creation Timestamp:  2021-12-08T19:35:26Z
  Finalizers:
    trident.netapp.io/astradsvolume-finalizer
    astrads.netapp.io/astradsvolume-finalizer
  Generation:  1
  Managed Fields:
    API Version:  astrads.netapp.io/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:labels:
          f:astrads.netapp.io/cluster:
          f:astrads.netapp.io/mip:
          f:astrads.netapp.io/volumeUUID:
      f:status:
        .:
        f:cluster:
        f:conditions:
        f:created:
        f:displayName:
        f:exportAddress:
        f:internalName:
        f:mip:
        f:permissions:
        f:qosPolicy:
        f:requestedSize:
        f:restoreCacheSize:
        f:size:
        f:snapshotReservePercent:
        f:state:
        f:volumePath:
        f:volumeUUID:
    Manager:      cluster-controller
    Operation:    Update
    Time:         2021-12-08T19:35:32Z
    API Version:  astrads.netapp.io/v1alpha1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        f:exportPolicy:
    Manager:      dms-controller
    Operation:    Update

```

```

Subresource:  status
Time:         2021-12-08T19:35:32Z
API Version:  astrads.netapp.io/v1alpha1
Fields Type:  FieldsV1
fieldsV1:
  f:metadata:
    f:annotations:
      .:
    f:provisioning:
    f:trident:
  f:finalizers:
    v:"trident.netapp.io/astradsvolume-finalizer":
  f:labels:
    .:
    f:trident.netapp.io/cloud:
    f:trident.netapp.io/creator:
    f:trident.netapp.io/performance:
  f:spec:
    .:
    f:cluster:
    f:displayName:
    f:exportPolicy:
    f:noSnapDir:
    f:permissions:
    f:qosPolicy:
    f:size:
    f:snapshotReservePercent:
    f:type:
    f:volumePath:
  Manager:      trident_orchestrator
  Operation:     Update
  Time:         2021-12-08T19:35:34Z
Resource Version: 12007115
UID:            d522ae4f-e793-49ed-bbe0-9112d7f9167b
Spec:
  Cluster:      jai-ads
  Display Name:  pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  Export Policy: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  No Snap Dir:  true
  Permissions:  0777
  Qos Policy:   silver
  Size:         9042036412
  Snapshot Reserve Percent: 5
  Type:         ReadWrite
  Volume Path:  /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Status:

```

```

Cluster:  jai-ads
Conditions:
  Last Transition Time:  2021-12-08T19:35:32Z
  Message:              Volume is online
  Reason:               VolumeOnline
  Status:               True
  Type:                 AstraDSVolumeOnline
  Last Transition Time:  2021-12-08T19:35:32Z
  Message:              Volume creation request was successful
  Reason:               VolumeCreated
  Status:               True
  Type:                 AstraDSVolumeCreated
Created:                true
Display Name:           pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Export Address:         10.192.2.192
Export Policy:           pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Internal Name:          pvc_1133453a_e2f5_48a5_a06c_d14b8aa7be07
Mip:                    10.192.1.192
Permissions:            777
Qos Policy:              silver
Requested Size:          9042036412
Restore Cache Size:      0
Size:                    8830116Ki
Snapshot Reserve Percent: 5
State:                   online
Volume Path:             /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Volume UUID:             cf33fd38-a451-596c-b656-61b8270d2b5e
Events:
  Type      Reason          Age    From                                Message
  ----      -
  Normal    VolumeCreated  3m9s   ADSClusterController  Volume creation
request was successful

```

Manage the Astra Data Store cluster

You can manage the cluster by using kubectl commands with Astra Data Store.

- [Add a node](#)
- [Remove a node](#)
- [Place a node in maintenance mode](#)
- [Add drives to a node](#)
- [Replace a drive](#)

What you'll need

- System with kubectl and kubectl-astrads plugin installed. See [Install Astra Data Store](#).

Add a node

The node that you are adding should be part of the Kubernetes cluster and should have a configuration that is similar to the other nodes in the cluster.



To add a node using Astra Control Center, see [Add nodes to a storage backend cluster](#).

Steps

1. If the new node's dataIP is not already part of the Astra Data Store cluster CR, do the following:
 - a. Edit the cluster CR and add the additional dataIP in the `adsDataNetworks` **addresses** field. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl edit astradscluster CLUSTER_NAME -n astrads-system
```

- b. Save the CR.
- c. Add the node to the Astra Data Store cluster. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

2. Otherwise, just add the nodes. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads nodes add --cluster CLUSTER_NAME
```

3. Verify that the node has been added:

```
kubectl astrads nodes list
```

Remove a node

Use `kubectl` commands with Astra Data Store to remove a node in a cluster.

Steps

1. List all the nodes:

```
kubectl astrads nodes list
```

Response:

| NODE NAME | NODE STATUS | CLUSTER NAME |
|--------------------|-------------|--------------------------|
| sti-rx2540-534d... | Added | cluster-multinodes-21209 |
| sti-rx2540-535d... | Added | cluster-multinodes-21209 |
| ... | | |

2. Mark the node for removal. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads nodes remove NODE_NAME
```

Response:

```
Removal label set on node sti-rx2540-534d.lab.org  
Successfully updated ADS cluster cluster-multinodes-21209 desired node  
count from 4 to 3
```

After you mark the node for removal, the node status should change from active to present.

3. Verify the present status of the removed node:

```
kubectl get nodes --show-labels
```

Response:

| NAME | STATUS | ROLES |
|---|--------|----------------------------|
| sti-astramaster-050.lab.org v1.20.0 beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-astramaster-050.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master= | Ready | control-plane,master 3h39m |
| sti-rx2540-556a.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true | Ready | worker 3h38m |
| sti-rx2540-556b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-556b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true | Ready | worker 3h38m |
| sti-rx2540-534d.lab.org v1.20.0 astrads.netapp.io/storage-cluster-status=present,astrads.netapp.io/storage-node-removal=,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557a.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true | Ready | worker 3h38m |
| sti-rx2540-557b.lab.org v1.20.0 astrads.netapp.io/cluster=astrads-cluster-890c32c,astrads.netapp.io/storage-cluster-status=active,beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=sti-rx2540-557b.lab.org,kubernetes.io/os=linux,node-role.kubernetes.io/worker=true | Ready | worker 3h38m |

- Uninstall Astra Data Store from the node. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads nodes uninstall NODE_NAME
```

- Verify the node is removed from the cluster:

```
kubectl astrads nodes list
```

Result

The node is removed from Astra Data Store.

Place a node in maintenance mode

When you need to perform host maintenance or package upgrades, you should place the node in maintenance mode.



The node must already be part of the Astra Data Store cluster.

When a node is in maintenance mode, you cannot add a node to the cluster. In this example, we will place node nhcitjj1525 into maintenance mode.

Steps

1. Display the node details:

```
kubectl get nodes
```

Response:

| NAME | STATUS | ROLES | AGE | VERSION |
|----------------|--------|----------------------|-------|---------|
| nhcitjj1525 | Ready | <none> | 3d18h | v1.23.5 |
| nhcitjj1526 | Ready | <none> | 3d18h | v1.23.5 |
| nhcitjj1527 | Ready | <none> | 3d18h | v1.23.5 |
| nhcitjj1528 | Ready | <none> | 3d18h | v1.23.5 |
| scs000039783-1 | Ready | control-plane,master | 3d18h | v1.23.5 |

2. Ensure that the node is not already in maintenance mode:

```
kubectl astrads maintenance list
```

Response (there are no nodes already in maintenance mode):

| NAME | NODE NAME | IN MAINTENANCE | MAINTENANCE STATE | MAINTENANCE VARIANT |
|------|-----------|----------------|-------------------|---------------------|
|------|-----------|----------------|-------------------|---------------------|

3. Enable maintenance mode. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads maintenance create CR_NAME --node-name=NODE_NAME  
--variant=Node
```

For example:

```
kubectl astrads maintenance create maint1 --node-name="nhcitjj1525"
--variant=Node
```

Response:

```
Maintenance mode astrads-system/maint1 created
```

4. List the nodes:

```
kubectl astrads nodes list
```

Response:

| NODE NAME | NODE STATUS | CLUSTER NAME |
|-------------|-------------|----------------|
| nhcitjj1525 | Added | ftap-astra-012 |
| ... | | |

5. Check the status of the maintenance mode:

```
kubectl astrads maintenance list
```

Response:

| NAME | NODE NAME | IN MAINTENANCE | MAINTENANCE STATE |
|---------------------|-------------|----------------|---------------------|
| MAINTENANCE VARIANT | | | |
| node4 | nhcitjj1525 | true | ReadyForMaintenance |

The In Maintenance mode starts as false and changes to true.

The Maintenance State changes from PreparingForMaintenance to ReadyforMaintenance.

6. After the node maintenance is complete, disable maintenance mode:

```
kubectl astrads maintenance update maint1 --node-name="nhcitjj1525"
--variant=None
```

7. Ensure that the node is no longer in maintenance mode:

```
kubectl astrads maintenance list
```

Add drives to a node

Use `kubectl` commands with Astra Data Store to add physical or virtual drives to a node in an Astra Data Store cluster.

What you'll need

- One or more drives that meet the following criteria:
 - Already installed in the node (physical drives) or added to the node VM (virtual drives)
 - No partitions on the drive
 - Drive is not currently in use by the cluster
 - Drive raw capacity does not exceed licensed raw capacity in the cluster (For example, with a license granting 2TB of storage per CPU core, a cluster of 10 nodes would have a maximum of 20TB raw drive capacity)
 - Drive is at least the size of other active drives in the node



Astra Data Store requires no more than 16 drives per node. If you try to add a 17th drive, the drive add request is denied.

Steps

1. Describe the cluster:

```
kubectl astrads clusters list
```

Response:

| CLUSTER NAME | CLUSTER STATUS | NODE COUNT |
|--------------------------|----------------|------------|
| cluster-multinodes-21209 | created | 4 |

2. Note the cluster name.
3. Show the drives that are available to add to all nodes in the cluster. Replace `CLUSTER_NAME` with the name of your cluster:

```
kubectl astrads adddrive show-available --cluster=CLUSTER_NAME
```

Response:

Current cluster drive add status
Licensed cluster capacity: 72.0 TiB
Cluster capacity used: 2.3 TiB
Maximum node size without stranding: 800.0 GiB

Node: node1.name
Current node size: 600.0 GiB
Maximum licensed node size: 18.0 TiB
Total size that can be added to this node without stranding: 200.0 GiB
Add drive minimum/recommended size: 100.0 GiB. Larger disks will be constrained to this size
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c290e16d52479a9af5eac c290e16d52479a9af5eac 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c2935798df68355dee0be c2935798df68355dee0be 0
100 GiB false

Node: node2.name
Current node size: 600.0 GiB
Maximum licensed node size: 18.0 TiB
Total size that can be added to this node without stranding: 200.0 GiB
Add drive minimum/recommended size: 100.0 GiB. Larger disks will be constrained to this size
No suitable drives to add exist.

Node: node3.name
Current node size: 600.0 GiB
Maximum licensed node size: 18.0 TiB
Total size that can be added to this node without stranding: 200.0 GiB
Add drive minimum/recommended size: 100.0 GiB. Larger disks will be constrained to this size
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c29ee82992ed7a36fc942 c29ee82992ed7a36fc942 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c29312aa362469fb3da9c c29312aa362469fb3da9c 0
100 GiB false

Node: node4.name
Current node size: 600.0 GiB
Maximum licensed node size: 18.0 TiB
Total size that can be added to this node without stranding: 200.0 GiB
Add drive minimum/recommended size: 100.0 GiB. Larger disks will be constrained to this size
No suitable drives to add exist.

4. Do one of the following:

- If all available drives have the same name, you can add them to the respective nodes simultaneously. Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads adddrive create --cluster=CLUSTER_NAME --name  
REQUEST_NAME --drivesbyname all=DRIVE_NAME
```

- If the drives are named differently, you can add them to the respective nodes one at a time (you'll need to repeat this step for each drive you need to add). Replace the information in capital letters with the appropriate values for your environment:

```
kubectl astrads adddrive create --cluster=CLUSTER_NAME --name  
REQUEST_NAME --drivesbyname NODE_NAME=DRIVE_NAME
```

Result

Astra Data Store creates a request to add the drive or drives, and a message appears with the result of the request.

Replace a drive

When a drive fails in a cluster, the drive must be replaced as soon as possible to ensure data integrity. If a drive fails, you can see information about the failed drive in cluster CR node status, cluster health condition information, and the metrics endpoint. You can use the following example commands to see failed drive information.

Example of cluster showing failed drive in `nodeStatuses.driveStatuses`

```
kubectl get adscl -A -o yaml
```

Response:

```
...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
...
nodeStatuses:
  - driveStatuses:
    - driveID: 31205e51-f592-59e3-b6ec-185fd25888fa
      driveName: scsi-36000c290ace209465271ed6b8589b494
      drivesStatus: Failed
    - driveID: 3b515b09-3e95-5d25-a583-bee531ff3f31
      driveName: scsi-36000c290ef2632627cb167a03b431a5f
      drivesStatus: Active
    - driveID: 0807fa06-35ce-5a46-9c25-f1669def8c8e
      driveName: scsi-36000c292c8fc037c9f7e97a49e3e2708
      drivesStatus: Active
  ...
```

Example of new AstraDSFailedDrive CR

The failed drive CR is created automatically in the cluster with a name corresponding to the UUID of the failed drive.

```
kubect1 get adsfd -A -o yaml
```

Response:

```
...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSFailedDrive
metadata:
  name: c290a-5000-4652c-9b494
  namespace: astrads-system
spec:
  executeReplace: false
  replaceWith: ""
status:
  cluster: arda-6e4b4af
  failedDriveInfo:
    failureReason: AdminFailed
    inUse: false
    name: scsi-36000c290ace209465271ed6b8589b494
    path: /dev/disk/by-id/scsi-36000c290ace209465271ed6b8589b494
    present: true
    serial: 6000c290ace209465271ed6b8589b494
    node: sti-rx2540-300b.lab.org
  state: ReadyToReplace
```

```
kubectl astrads faileddrive list --cluster arda-6e4b4af
```

Response:

| NAME | NODE | CLUSTER | STATE |
|----------|--------------------------------|-------------|----------------|
| AGE | | | |
| 6000c290 | sti-rx2540-300b.lab.netapp.com | ard-6e4b4af | ReadyToReplace |
| 13m | | | |

Steps

1. List possible replacement drives with the `kubectl astrads faileddrive show-replacements` command, which filters drives that fit replacement restrictions (unused in cluster, not mounted, no partitions, and equal or larger than failed drive).

To list all drives without filtering possible replacement drives, add `--all` to `show-replacements` command.

```
kubectl astrads faileddrive show-replacements --cluster ard-6e4b4af
--name 6000c290
```

Response:

| NAME | IDPATH | SERIAL | PARTITIONCOUNT | MOUNTED | SIZE |
|------|-------------------|--------|----------------|---------|-------|
| sdh | /scsi-36000c29417 | 45000c | 0 | false | 100GB |

2. Use the `replace` command to replace the drive with the passed serial number. The command completes the replacement or fails if `--wait` time elapses.

```
kubectl astrads faileddrive replace --cluster arda-6e4b4af --name
6000c290 --replaceWith 45000c --wait
```

Response:

```
Drive replacement completed successfully
```



If `kubectl astrads faileddrive replace` is executed using an inappropriate `--replaceWith` serial number, an error appears similar to this:

```
kubectl astrads replacedrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith BAD_SERIAL_NUMBER
Drive 6000c2927 replacement started
Failed drive 6000c2927 has been set to use BAD_SERIAL_NUMBER as a
replacement
...
Drive replacement didn't complete within 25 seconds
Current status: {FailedDriveInfo:{InUse:false Present:true Name:scsi-
36000c2 FiretapUUID:444a5468 Serial:6000c Path:/scsi-36000c
FailureReason:AdminFailed Node:sti-b200-0214a.lab.netapp.com}
Cluster:astrads-cluster-f51b10a State:ReadyToReplace
Conditions:[{Message: "Replacement drive serial specified doesn't
exist", Reason: "DriveSelectionFailed", Status: False, Type:' Done'}]}
```

3. To re-run drive replacement use `--force` with the previous command:

```
kubectl astrads faileddrive replace --cluster astrads-cluster-f51b10a
--name 6000c2927 --replaceWith VALID_SERIAL_NUMBER --force
```

For more information

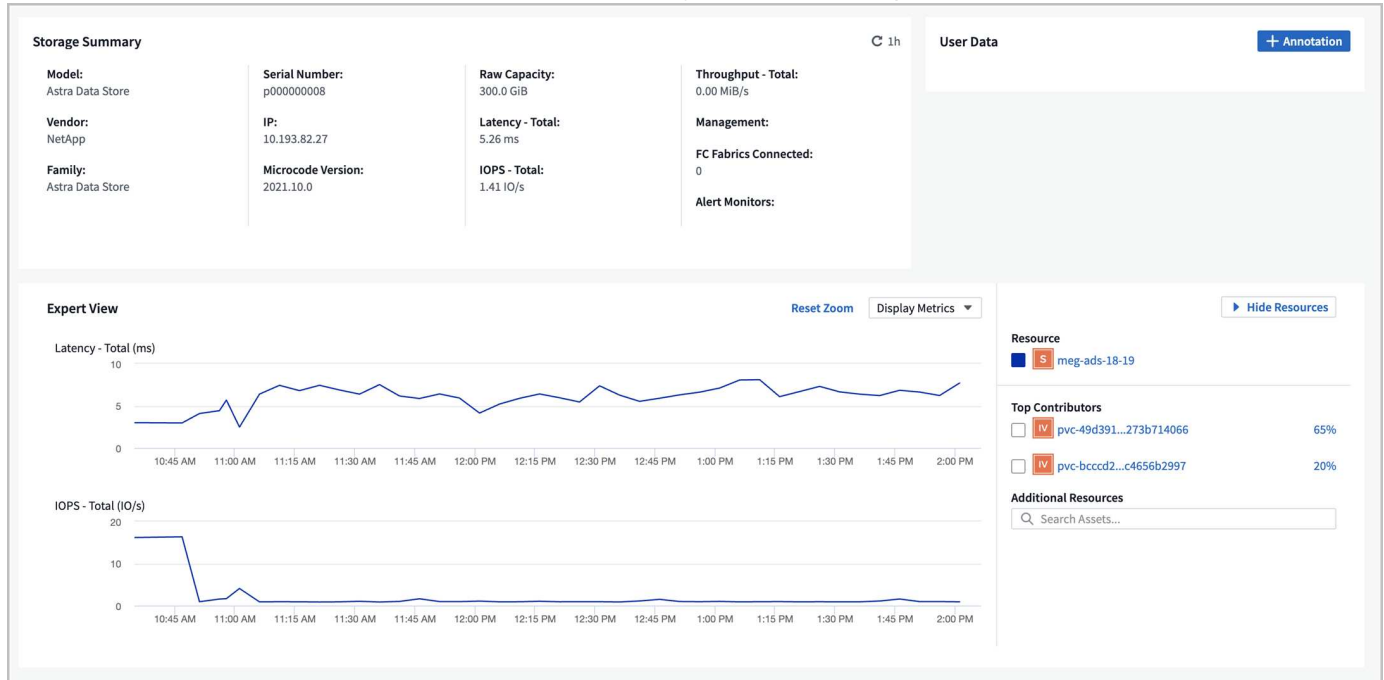
- [Manage Astra Data Store resources with kubectl commands](#)
- [Add nodes to a storage backend cluster in Astra Control Center](#)

Monitor Astra Data Store

Monitor metrics with Cloud Insights

You can monitor Astra Data Store metrics using Cloud Insights.

Here are some sample Astra Data Store metrics displayed in Cloud Insights:



You can also display a list of metrics generated in Astra Data Store by using the [Open Metrics API help](#).

You can complete the following tasks:

- [Complete Cloud Insights connection prerequisite tasks](#)
- [Acquisition Unit storage](#)
- [Download and run the installation script](#)
- [Edit the Cloud Insights connection](#)
- [Disconnect from Cloud Insights](#)

Complete Cloud Insights connection prerequisite tasks

Prior to connecting Astra Data Store with Cloud Insights, you need to complete these tasks:

- [Install the Astra Data Store Monitoring Operator](#) that is part of the Astra Data Store installation instructions.
- [Install the kubecti-astrads binary](#) that is part of the Astra Data Store installation instructions.
- [Create a Cloud Insights account](#).
- Ensure that the following commands are available: `awk`, `curl`, `grep` and `jq`

Gather the following information:

- **Cloud Insights API access token** with Read/Write permissions to the categories: Acquisition Unit, Data Collection, Data Ingestion and Log Ingestion. This will be used for the read/write operations, setting up the

Acquisition Unit, and setting up data ingest processes.

- **Kubernetes API server IP address and port.** This is used to monitor the Astra Data Store cluster.
- **Kubernetes API token.** This is used to call Kubernetes APIs.
- **Persistent volume configuration.** Information about how persistent volumes are provisioned.

Acquisition Unit storage

The Acquisition Unit requires three persistent volumes for storing installation files, configuration data and logs. The Monitoring Operator uses the default storage class to create persistent volume claims. You can specify a different storage class name using the `-s` option when running the installer script.

If your Kubernetes cluster does not have a storage provisioner (such as NetApp Trident), you can provide a local filesystem path using the `-r` option when running the installer script. When the `-r` option is set, the installer script creates three persistent volumes inside the provided directory. This directory requires a minimum of 150 GB free space.

Download and run the installation script

Cloud Insights provides a Bash script to enable Astra Data Store monitoring via the Monitoring Operator. The install script will install an Acquisition Unit with the Astra Data Store collector and a Fluent Bit agent.

The Cloud Insights tenant domain name and selected Cloud Insights API access token will be embedded in the installer script when it is downloaded.

Then, metrics will be sent as follows:

- The Cloud Insights Acquisition Unit will send metrics to the Cloud Insights data lake.
- Fluent Bit will send logs to the log ingestion service.

Display installer script help

The full help text for the installer script is shown below:

Display installer script help text:

```
./cloudinsights-ads-monitoring.sh -h
```

Response:

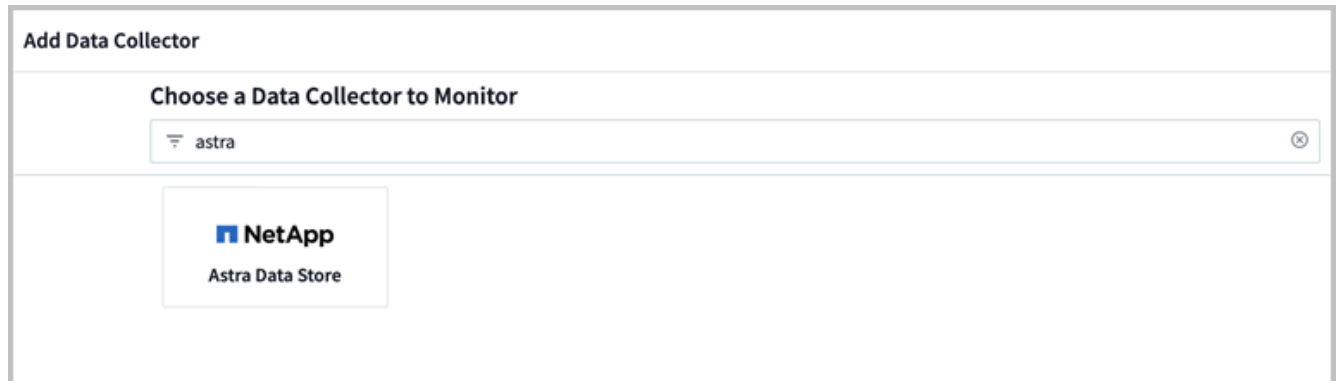
```

USAGE: cloudinsights-ads-monitoring.sh [OPTIONS]
Configure monitoring of Astra Data Store by Cloud Insights.
OPTIONS:
  -h                                Display this help message.
  -d ci_domain_name                 Cloud Insights tenant domain name.
  -i kubernetes_ip                  Kubernetes API server IP address.
  -k ci_api_key                     Cloud Insights API Access Token.
  -n namespace                      Namespace for monitoring components. (default:
netapp-monitoring)
  -p kubernetes_port                Kubernetes API server port. (default: 6443)
  -r root_pv_dir                    Create 3 Persistent Volumes in this directory
for the Acquisition Unit.
                                   Only specify this option if there is no Storage
Provisioner installed and the PVs do not already exist.
  -s storage_class                  Storage Class name for provisioning Acquisition
Unit PVs. If not specified, the default storage class will be used.
  -t kubernetes_token               Kubernetes API server token.

```

Run the install script

1. Create a Cloud Insights account, if you don't already have one.
2. Log in to Cloud Insights.
3. From the Cloud Insights menu, click on **Admin > Data Collectors**.
4. Click on **+ Data Collector** to add a new collector.



5. Click on the **Astra Data Store** tile.
6. Select the correct Cloud Insights API access token or create a new one.
7. Follow the instructions to download the installer script, update the permissions, and run the script.

The script contains your Cloud Insights tenant URL and the selected Cloud Insights API access token.



Select a Data Collector



Configure Collector



NetApp
Astra Data Store

Configure Collector

Configure Kubernetes Operator to monitor NetApp Astra Data Store (ADS).

What Operating System or Platform Are You Using?

 Kubernetes

Select existing API Access Token or create a new one

default_ads_api_key1 (...d0gHof)

[+ API Access Token](#)

[Production Best Practices ?](#)

Configure Astra Data Store [Need Help?](#)

- 1

The commands *awk*, *curl*, *grep*, *jq*, *kubectl* and the *kubectl-astads* plugin must be installed where the installer script is run. You will need the Kubernetes API server IP address and a Kubernetes API token to run this install script. See the documentation if you need help finding this information.
- 2

Copy Installer Script

☐ Reveal Installer Script

```
#!/usr/bin/env bash
SCRIPT='basename $0'

CI_DOMAIN_NAME="f49uaky.gstabler-ads.cloudinsights-dev.netapp.com"
CI_API_KEY="eyJraWQ1OiI5OTk5IiwidHlwIjoI5ldUIiwiaWxnIjoI5FMzODQifQ.eyJjcVhdG9yTG9naW41OiJhZG1pb2IiImRpc3BsYXl0YW11IjoIZGVmYXVsdF9hZHNfYXBPX2tleTEgKG9uIGJlaGFsZiBvZiBhZG1pb2I"

```
- 3

Copy the above installer script and save it as *cloudinsights-ads-monitoring.sh*
- 4

Copy Permissions Command

☐ Reveal Permissions Command

```
chmod +x cloudinsights-ads-monitoring.sh
```
- 5

Paste the permissions command in a terminal to enable execute permissions on the installer script.
- 6

Copy Install Command

☐ Reveal Install Command

```
./cloudinsights-ads-monitoring.sh -i <KUBERNETES_IP> -t <KUBERNETES_TOKEN>
```
- 7

Paste the install command in a terminal, replace the placeholders with the correct values for your environment, and run the command. It will take several minutes to complete. The script will use the default namespace 'netapp-monitoring'. Additional options are available for customized environments. The default configuration for kubectl should point to the kubernetes cluster to be monitored.
- 8

Complete Setup

8. Click **Complete Setup** after the script completes.

After the installation script completes, the Astra Data Store collector appears in the Datasources list.



If the script exits due to an error, you can run it again later once the error is resolved. The script supports additional parameters such as the Monitoring Operator namespace and Kubernetes API server port if your environment does not use the default settings. Use the `-h` option in `./cloudinsights-ads-monitoring.sh -h` to see the usage and help text.

The installation script produces output that looks like this when the configuration is successful:

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Acquisition Unit
...
Acquisition Unit has been installed successfully.
Configuring Astra Data Store data collector
Astra Data Store collector data '<CLUSTER_NAME>' created
Configuration done!
```

Example Agent CR

Below is an example of what the `monitoring-netapp` agent CR will look like after running the installer script.

```

spec:
  au:
    isEnabled: true
    storageClassName: auto-sc
  cluster-name: meg-ads-21-22-29-30
  docker-repo: docker.repo.eng.netapp.com/global/astra
  fluent-bit:
    - name: ads-tail
      outputs:
        - sink: ADS_STDOUT
      substitutions:
        - key: TAG
          value: firetapems
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
    - name: agent
    - name: ads-tail-ci
      outputs:
        - sink: CI
      substitutions:
        - key: TAG
          value: netapp.ads
        - key: LOG_FILE
          values:
            - /var/log/firetap/*/ems/ems
            - /var/log/firetap/ems/*/ems/ems
        - key: ADS_CLUSTER_NAME
          value: meg-ads-21-22-28-29-30
  output-sink:
    - api-key: abcd
      domain-name: bz19ngz.gst-adsdemo.ci-dev.netapp.com
      name: CI
  serviceAccount: sa-netapp-monitoring
  status:
    au-pod-status: UP
    au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9

```

Edit the Cloud Insights connection

You can later edit the Kubernetes API token or the Cloud Insights API access token:

- If you want to update Kubernetes API token, you should edit the Astra Data Store collector from the Cloud Insights UI.
- If you want to update the Cloud Insights API access token used for telemetry and logs, you should edit the Monitoring Operator CR using kubectl commands.

Update the Kubernetes API token

1. Log in to Cloud Insights.
2. Select **Admin > Data Collectors** to access the Data Collectors page.
3. Find the entry for the Astra Data Store cluster.
4. Click on the menu on the right side of the page, and select **Edit**.
5. Update the Kubernetes API Token field with the new value.
6. Select **Save Collector**.

Update the Cloud Insights API access token

1. Log in to Cloud Insights.
2. Create a new Cloud Insights API access token by selecting **Admin > API Access** and clicking **+API Access Token**.
3. Edit the Agent CR:

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

4. Locate the `output-sink` section and find the entry with the name `CI`.
5. For the label `api-key`, replace the current value with the new Cloud Insights API access token.

The section looks something like this:

```
output-sink:
  - api-key: <api key value>
    domain-name: <tenant url>
    name: CI
```

6. Save and quit the editor window.

The Monitoring Operator will update Fluent Bit to use the new Cloud Insights API access token.

Disconnect from Cloud Insights

To disconnect from Cloud Insights, you will need to delete the Astra Data Store collector from the Cloud Insights UI first. After that is complete, you can remove the Acquisition Unit, Telegraf (if configured) and Fluent Bit configurations from the Monitoring Operator.

Remove the Astra Data Store collector

1. Log in to Cloud Insights.

2. Select **Admin > Data Collectors** to access the Data Collectors page.
3. Find the entry for the Astra Data Store cluster.
4. Select the menu on the right side of the screen, and select **Delete**.
5. Click **Delete** on the confirmation page.

Remove the Acquisition Unit, Telegraf (if configured) and Fluent Bit

1. Edit the Agent CR:

```
kubectl --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

2. Locate the `au` section and set `isEnabled` to `false`
3. Locate the `fluent-bit` section and remove the plugin named `ads-tail-ci`. If there are no more plugins, you can remove the `fluent-bit` section.
4. If Telegraf is configured, locate the `telegraf` section and remove the plugin named `ads-open-metric`. If there are no more plugins, you can remove the `telegraf` section.
5. Locate the `output-sink` section and remove the sink named `CI`.
6. Save and quit the editor window.

The Monitoring Operator will update the Telegraf (if configured) and Fluent Bit configurations and delete the Acquisition Unit pod.

7. If you used local directories for the Acquisition Unit PVs instead of a Storage Provisioner, delete the PVs:

```
kubectl delete pv au-lib au-log au-pv
```

Then, delete the actual directories on the node where the Acquisition Unit was running.

8. After the Acquisition Unit pod has been deleted, you can delete the Acquisition Unit from Cloud Insights.
 - a. In the Cloud Insights menu, select **Admin > Data Collectors**.
 - b. Click on the **Acquisition Units** tab.
 - c. Click on the menu next to the Acquisition Unit pod.
 - d. Select **Delete**.

The Monitoring Operator updates the Telegraf (if configured) and Fluent Bit configurations and removes the Acquisition Unit.

Open Metrics API help

Here is a list of APIs that you can use to gather metrics from Astra Data Store.

- The "HELP" line describes the metric.
- The "TYPE" line indicates whether the metric is a gauge or a counter.


```

# HELP astrads_cluster_capacity_logical_percent Percentage cluster logical
capacity that is used (0-100)
# TYPE astrads_cluster_capacity_logical_percent gauge
# HELP astrads_cluster_capacity_max_logical Max Logical capacity of the
cluster in bytes
# TYPE astrads_cluster_capacity_max_logical gauge
# HELP astrads_cluster_capacity_max_physical The sum of the space in the
cluster in bytes for storing data after provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_max_physical gauge
# HELP astrads_cluster_capacity_ops The IO operations capacity of the
cluster
# TYPE astrads_cluster_capacity_ops gauge
# HELP astrads_cluster_capacity_physical_percent The percentage of cluster
physical capacity that is used (0-100)
# TYPE astrads_cluster_capacity_physical_percent gauge
# HELP astrads_cluster_capacity_used_logical The sum of the bytes of data
in all volumes in the cluster before provisioning efficiencies, data
reduction algorithms and replication schemes are applied
# TYPE astrads_cluster_capacity_used_logical gauge
# HELP astrads_cluster_capacity_used_physical Used Physical capacity of a
cluster in bytes
# TYPE astrads_cluster_capacity_used_physical gauge
# HELP astrads_cluster_other_latency The sum of the accumulated latency in
seconds for other IO operations of all the volumes in a cluster. Divide by
astrads_cluster_other_ops to get the average latency per other operation
# TYPE astrads_cluster_other_latency counter
# HELP astrads_cluster_other_ops The sum of the other IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_other_ops counter
# HELP astrads_cluster_read_latency The sum of the accumulated latency in
seconds of read IO operations of all the volumes in a cluster. Divide by
astrads_cluster_read_ops to get the average latency per read operation
# TYPE astrads_cluster_read_latency counter
# HELP astrads_cluster_read_ops The sum of the read IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_read_ops counter
# HELP astrads_cluster_read_throughput The sum of the read throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_read_throughput counter
# HELP astrads_cluster_storage_efficiency Efficacy of data reduction
technologies. (logical used / physical used)
# TYPE astrads_cluster_storage_efficiency gauge
# HELP astrads_cluster_total_latency The sum of the accumulated latency in
seconds of all IO operations of all the volumes in a cluster. Divide by
astrads_cluster_total_ops to get average latency per operation

```

```

# TYPE astrads_cluster_total_latency counter
# HELP astrads_cluster_total_ops The sum of the IO operations of all the
volumes in a cluster
# TYPE astrads_cluster_total_ops counter
# HELP astrads_cluster_total_throughput The sum of the read and write
throughput of all the volumes in a cluster in bytes
# TYPE astrads_cluster_total_throughput counter
# HELP astrads_cluster_utilization_factor The ratio of the current cluster
IO operations based on recent IO sizes to the cluster iops capacity. (0.0
- 1.0)
# TYPE astrads_cluster_utilization_factor gauge
# HELP astrads_cluster_volume_used The sum of used capacity of all the
volumes in a cluster in bytes
# TYPE astrads_cluster_volume_used gauge
# HELP astrads_cluster_write_latency The sum of the accumulated latency in
seconds of write IO operations of all the volumes in a cluster. Divide by
astrads_cluster_write_ops to get the average latency per write operation
# TYPE astrads_cluster_write_latency counter
# HELP astrads_cluster_write_ops The sum of the write IO operations of all
the volumes in a cluster
# TYPE astrads_cluster_write_ops counter
# HELP astrads_cluster_write_throughput The sum of the write throughput of
all the volumes in a cluster in bytes
# TYPE astrads_cluster_write_throughput counter
# HELP astrads_disk_base_seconds Base for busy, pending and queued.
Seconds since collection began
# TYPE astrads_disk_base_seconds counter
# HELP astrads_disk_busy Seconds the disk was busy. 100 *
(astrads_disk_busy / astrads_disk_base_seconds) = percent busy (0-100)
# TYPE astrads_disk_busy counter
# HELP astrads_disk_capacity Raw Capacity of a disk in bytes
# TYPE astrads_disk_capacity gauge
# HELP astrads_disk_io_pending Summation of the count of pending io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average pending operation count
# TYPE astrads_disk_io_pending counter
# HELP astrads_disk_io_queued Summation of the count of queued io
operations for a disk times time. Divide by astrads_disk_base_seconds to
get the average queued operations count
# TYPE astrads_disk_io_queued counter
# HELP astrads_disk_read_latency Total accumulated latency in seconds for
disk reads. Divide by astrads_disk_read_ops to get the average latency per
read operation
# TYPE astrads_disk_read_latency counter
# HELP astrads_disk_read_ops Total number of read operations for a disk
# TYPE astrads_disk_read_ops counter

```

```

# HELP astrads_disk_read_throughput Total bytes read from a disk
# TYPE astrads_disk_read_throughput counter
# HELP astrads_disk_write_latency Total accumulated latency in seconds for
disk writes. Divide by astrads_disk_write_ops to get the average latency
per write operation
# TYPE astrads_disk_write_latency counter
# HELP astrads_disk_write_ops Total number of write operations for a disk
# TYPE astrads_disk_write_ops counter
# HELP astrads_disk_write_throughput Total bytes written to a disk
# TYPE astrads_disk_write_throughput counter
# HELP astrads_value_scrape_duration Duration to scrape values
# TYPE astrads_value_scrape_duration gauge
# HELP astrads_volume_capacity_available The minimum of the available
capacity of a volume and the available capacity of the cluster in bytes
# TYPE astrads_volume_capacity_available gauge
# HELP astrads_volume_capacity_available_logical Logical available
capacity of a volume in bytes
# TYPE astrads_volume_capacity_available_logical gauge
# HELP astrads_volume_capacity_percent Percentage of volume capacity
available (0-100). (capacity available / provisioned) * 100
# TYPE astrads_volume_capacity_percent gauge
# HELP astrads_volume_capacity_provisioned Provisioned capacity of a
volume in bytes after setting aside the snapshot reserve. (size - snapshot
reserve = provisioned)
# TYPE astrads_volume_capacity_provisioned gauge
# HELP astrads_volume_capacity_size Total capacity of a volume in bytes
# TYPE astrads_volume_capacity_size gauge
# HELP astrads_volume_capacity_snapshot_reserve_percent Snapshot reserve
percentage of a volume (0-100)
# TYPE astrads_volume_capacity_snapshot_reserve_percent gauge
# HELP astrads_volume_capacity_snapshot_used The amount of volume snapshot
data that is not in the active file system in bytes
# TYPE astrads_volume_capacity_snapshot_used gauge
# HELP astrads_volume_capacity_used Used capacity of a volume in bytes.
This is bytes in the active filesystem unless snapshots are consuming more
than the snapshot reserve. (bytes in the active file system + MAX(0,
snapshot_used-(snapshot_reserve_percent/100*size))
# TYPE astrads_volume_capacity_used gauge
# HELP astrads_volume_other_latency Total accumulated latency in seconds
for operations on a volume that are neither read or write. Divide by
astrads_volume_other_ops to get the average latency per other operation
# TYPE astrads_volume_other_latency counter
# HELP astrads_volume_other_ops Total number of operations for a volume
that are neither read or write
# TYPE astrads_volume_other_ops counter
# HELP astrads_volume_read_latency Total accumulated read latency in

```

```

seconds for a volume. Divide by astrads_volume_read_ops to get the average
latency per read operation
# TYPE astrads_volume_read_latency counter
# HELP astrads_volume_read_ops Total number of read operations for a
volume
# TYPE astrads_volume_read_ops counter
# HELP astrads_volume_read_throughput Total read throughput for a volume
in bytes
# TYPE astrads_volume_read_throughput counter
# HELP astrads_volume_total_latency Total accumulated latency in seconds
for all operations on a volume. Divide by astrads_volume_total_ops to get
the average latency per operation
# TYPE astrads_volume_total_latency counter
# HELP astrads_volume_total_ops Total number of operations for a volume
# TYPE astrads_volume_total_ops counter
# HELP astrads_volume_total_throughput Total throughput for a volume in
bytes
# TYPE astrads_volume_total_throughput counter
# HELP astrads_volume_write_latency Total accumulated write latency in
seconds for volume. Divide by astrads_volume_write_ops to get the average
latency per write operation
# TYPE astrads_volume_write_latency counter
# HELP astrads_volume_write_ops Total number of write operations for a
volume
# TYPE astrads_volume_write_ops counter
# HELP astrads_volume_write_throughput Total write throughput for a volume
in bytes
# TYPE astrads_volume_write_throughput counter

```

Monitor metrics with Prometheus and Grafana

You can monitor Astra Data Store metrics with Prometheus and Grafana. You can configure Prometheus to gather metrics from the Astra Data Store Kubernetes cluster metrics endpoint, and you can use Grafana to visualize the metrics data.

What you'll need

- Make sure that you have downloaded and installed the Prometheus and Grafana packages on the Astra Data Store cluster or a different cluster that can communicate with the Astra Data Store cluster. Follow the instructions in the official documentation to install each tool:
 - [Install Prometheus](#)
 - [Install Grafana](#)
- Prometheus and Grafana need to be able to communicate with the Astra Data Store Kubernetes cluster. If Prometheus and Grafana are not installed on the Astra Data Store cluster, you need to make sure they can communicate with the metrics service running on the Astra Data Store cluster.

Configure Prometheus

Astra Data Store exposes a metrics service on TCP port 9341 in the Kubernetes cluster. You need to configure Prometheus to collect metrics from this service.

Steps

1. Edit the `prometheus.yml` configuration file for your Prometheus installation.
2. Add a service target that points to the Astra Data Store service name and its port. For example:

```
scrape_configs:
static_configs:
- targets: ['astrads-metrics-service.astrads-system:9341']
```

3. Start the Prometheus service.

Configure Grafana

You can configure Grafana to display the metrics collected by Prometheus.

Steps

1. Edit the `datasources.yml` configuration file for your Grafana installation.
2. Add Prometheus as a data source. For example:

```
apiVersion: 1

datasources:
- name: astradatastore-prometheus
  type: prometheus
  access: proxy
  url: http://localhost:9090
  jsonData:
    manageAlerts: false
```

3. Start the Grafana service.
4. Follow the instructions in the Grafana documentation to [get started](#).

Import Grafana dashboard templates

The bundle file you downloaded to install Astra Data Store includes Grafana dashboard template files that you can import from within Grafana. These dashboard templates can help you see the types of metrics that are available from Astra Data Store and how you can view them.

Steps

1. Open the Astra Data Store `.tar.gz` bundle.
2. Open the `manifests` directory.
3. Extract the `grafana_cluster.json` and `grafana_volume.json` files.

4. Using the Grafana web UI, [import the dashboard template files in to Grafana](#).

Configure and monitor event logs

To monitor Event Management System (EMS) logs, you can do the following high level tasks:

- [Configure monitoring in the Astra Data Store cluster custom resource \(CR\)](#)
- [Set up Cloud Insights](#)
- [Stream event logs to Elastic](#).

Configure monitoring in the Astra Data Store cluster custom resource (CR)

If the monitoring option has not been configured on the Astra Data Store cluster CR, you can set it up using `astrads` extensions.

Enter:

```
kubectl astrads monitoring setup -n <NAMESPACE OF AGENT INSTALLED> -r  
<DOCKER REPO TO FIND FLUENT/TELEGRAF ETC IMAGES>
```

where:

- Namespace of agent installed: Enter the namespace for the Monitoring agent, which is the default name of the monitoring-netapp CR for the Monitoring Operator.
- `-r` is optional to set up the Docker registry where the Fluent or Telegraf images are located. By default, the path is set to `docker.repo.eng.netapp.com/global/astra`, which you can change.

Set up Cloud Insights

To view the logs, setting up Cloud Insights is optional; however, it is helpful to view data using Cloud Insights. See [how to set up NetApp Cloud Insights](#) for use with Astra Data Store.

Stream event logs to Elastic

To stream EMS events and other pod logs to a third-party endpoint such as Elastic, use the `astrads` extensions.

Enter:

```
kubectl astrads monitoring --host <ELASTIC HOST NAME> --port <ELASTIC HOST  
PORT> es
```



The Elastic host name can be an IP address.

Secure Astra Data Store

Manage security certificates

Astra Data Store uses Mutual Transport Layer Security (mTLS) encryption between the cluster's software components. Each Astra Data Store cluster has a self-signed root CA certificate (`astrads-cert-root`) and an intermediate CA certificate (`astrads-cert-<cluster_name>`).

These certificates are managed by the Astra Data Store operator; the operator automatically renews each certificate 7 days before their expiration date. You can also revoke the certificates manually.

Revoke a certificate

If an Astra Data Store controller, node, or CA certificate is compromised, you can revoke it by deleting its mTLS secret. When you do this, the Astra Data Store operator automatically issues a new certificate. You can revoke an Astra Data Store certificate at any time.



If you revoke a CA certificate, this will revoke any certificates signed by that CA.

Steps

1. Log in to the controller node in the Astra Data Store cluster.
2. List the existing certificates on the system. For example:

```
kubectl get secrets -n astrads-system | grep astrads-cert
```

The output should be similar to the following:

```
astrads-cert-astrads-cluster-controller
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-dms-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-ds-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-astrads-support-astrads-cluster-f23d158
kubernetes.io/tls      4      6d6h
astrads-cert-root
kubernetes.io/tls      4      6d6h
astrads-cert-sti-net-com
kubernetes.io/tls      5      6d6h
```

3. In the output, note the name of the certificate you need to revoke.
4. Use the `kubectl` utility to revoke the certificate, replacing `CERTIFICATE_NAME` with the name of the certificate. For example:

```
kubectl delete secret CERTIFICATE_NAME -n astrads-system
```

Result

The existing certificate is revoked, and a new certificate is automatically generated in its place.

Manage external keys

You can use one or more external key management servers to secure the keys that the cluster uses to access encrypted data. An external key management server is a third-party system in your storage environment that serves keys to nodes using the Key Management Interoperability Protocol (KMIP).



Astra Data Store enables Software Encryption at Rest (SEAR) with an internal key provider by default when the Astra Data Store cluster is created.

Managing keys involves the following custom resource definitions (CRDs):

- **AstraDSKeyProvider**: Configures an external KMIP server, which could be a cluster of servers.
- **AstraDSSEARKeyRotate**: Gets a new key encryption key from the key provider and provides it to Astra Data Store.

You can perform the following tasks related to external key management:

- [Set up external key management](#)
- [Check the software encryption at rest status](#)
- [Change external to internal key management](#)
- [Rotate keys for security](#)

Set up external key management

Setting up external key management in Astra Data Store uses `kubectl astrads` commands.

What you'll need

You will need an SSL certificate on the cluster or KMIP server that enables you to set up external keys, for example, by using OpenSSL.

Steps

1. Prepare the certificate for the key provider client. Include the client certificate, client private key, and trust CA bundles.



You'll prepare the SSL certificate on the cluster or KMIP server that allows you to set up external keys, for example, by using OpenSSL.

2. Log in to one of the nodes in the Astra Data Store cluster.
3. Configure the key provider for the Astra Data Store cluster by entering the following `kubectl` extension command:

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=<kmip_server_ip> <key_provider_cr_name>
--namespace astrads-system --cluster <ads_cluster_name>
```


Example

The following example configures an external key provider named "hashicorp" for ADS Cluster "astrads-cluster-f23d158".

```
kubectl-astrads key-provider certs --key key.pem
--client-cert client_cert.pem --ca-cert server_ca.pem
--hostnames=10.235.nnn.nnn hashicorp
--namespace astrads-system --cluster astrads-cluster-f23d158
```

1. Configure the Astra Data Store cluster to use an external key manager for SEAR via the AstraDSCluster CR. Display the help.

```
kubectl-astrads clusters sears -h
```

Response:

Configure SEARS in AstraDS cluster

Usage:

```
astrads clusters sears [flags]
```

Flags:

```
-d, --duration string    Duration for key rotation (default "2160h")
-h, --help               help for sears
```

Global Flags:

```
--ads-cluster-name string      Name of the ADS Cluster
--ads-cluster-namespace string Namespace of the ADS Cluster
...
```

Example

The following command configures the Astra Data Store cluster to use `AstraDSKeyProvider hashicorp` as the key manager of SEAR. The command also uses the key rotate time, which has the default value of 90 days (2160 hours).

```
kubectl-astrads clusters sears -d 500h hashicorp
--ads-cluster-name=astrads-cluster-f23d158
--ads-cluster-namespace=astrads-system
```

Check the software encryption at rest status

You can check the configuration of the software encryption at rest.

Step

1. Inspect the AstraDSCluster CR.

```
Name:          astrads-cluster-f23d158
Namespace:     astrads-system
Labels:        <none>
Annotations:   <none>
API Version:   astrads.netapp.io/v1beta1
Kind:          AstraDSCluster
...
Spec:
...
  Software Encryption At Rest:
    Ads Key Provider:      hashicorp
    Key Rotation Period:   500h0m0s
...
Status:
...
  Software Encryption At Rest Status:
    Key Active Time:       2022-05-16T15:53:47Z
    Key Provider Name:     hashicorp
    Key Provider UUID:     ccfc2b0b-dd98-5ca4-b778-99debef83550
    Key UUID:              nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnnn
```

Change external to internal key management

If you are currently using an external key manager, you can change it to an internal key manager.

Steps

1. Change the AstraDSCluster CR by removing the SoftwareEncryptionAtRest configuration.
2. (Optional) Delete the previous AstraDSKeyProvider and associated secret.



The previous key provider and secret will not be removed automatically.

Rotate keys for security

Key rotation enhances security. By default, Astra Data Store rotates keys automatically every 90 days. You can change the default setting. Additionally, you can rotate keys on demand when you want.

Configure automatic key rotation

1. Update the AstraDSSEARKeyRotate parameter in the CRD.

```
kubectl patch astradscluster astrads-cluster-f23d158
-n astrads-system
--type=merge -p '{"spec": {"softwareEncryptionAtRest": {
"keyRotationPeriod": "3000h"}}}'
```

Configure on-demand key rotation

1. Create an AstraDSSEARKeyRotateRequest CR to rotate keys.

```
cat << EOF | kubectl apply -f -
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSSEARKeyRotateRequest
metadata:
  name: manual
  namespace: astrads-system
spec:
  cluster: astrads-cluster-f23d158
EOF
```

Update the Astra Data Store license

You can update the installed evaluation license for Astra Data Store to extend the evaluation period. You can update the license using one of three methods:

- To update the Astra Data Store license using Astra Control Center, see [Update a storage backend license](#).
- To update the Astra Data Store license using the Astra VMware Plugin, see [Manage Astra Data Store with VMware](#).
- To update the Astra Data Store license using the command line, see [Update the Astra Data Store license using the command line](#).

Update the Astra Data Store license using the command line

You can use the `kubectl` utility to update the Astra Data Store license.

Steps

1. Apply the replacement 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>`) 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
```

You should see a response similar to the following:

| NAME | ADSCUSTER | VALID | PRODUCT |
|------------|-------------------------|-----------|------------------|
| EVALUATION | ENDDATE | VALIDATED | |
| p100000006 | astrads-example-cluster | true | Astra Data Store |
| 2023-01-23 | 2022-04-04T14:38:54Z | | true |

Upgrade Astra Data Store

You can upgrade Astra Data Store to take advantage of the latest features and fixes. You can upgrade Astra Data Store by using the Astra Data Store `kubectl` extension.

Upgrade Astra Data Store using kubectl

You can use the Astra Data Store `kubectl` extension to upgrade Astra Data Store.

Download the Astra Data Store bundle and extract the images

Steps

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

Steps

1. Copy the kubectl-astrads binary from the directory you used to extract images to the standard path where Kubernetes 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 following 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(s): ~~')
    astraImageShort=`echo $astraImage | sed 's~.*/~~'`
    docker tag ${astraImage} ${REGISTRY}/${astraImageShort}
    docker push ${REGISTRY}/${astraImageShort}
done
sed -i 's~\[YOUR_REGISTRY\]~'\${REGISTRY}'~'
./astrads/manifests/*.yaml
```

Perform the upgrade

Steps

1. Copy the `astradsoperator.yaml` file to the local directory:

```
cp /PATH/TO/FILE/astradsoperator.yaml ./
```

2. Upgrade the operator. Replace arguments in capital letters with the appropriate information for your environment:

```
kubectl-astrads upgrade ads-operator --repository-url REPOSITORY_URL  
--operator-yaml astradsoperator.yaml
```

3. Begin the Astra Data Store upgrade. Replace arguments in capital letters with the appropriate information for your environment:

```
kubectl-astrads upgrade ads-version --repository-url REPOSITORY_URL  
--ads-version-yaml ./astrads/manifests/astradsversion.yaml
```

A message appears notifying you that the upgrade has started, and will take a few minutes to complete.

Uninstall Astra Data Store with an automated script

To uninstall Astra Data Store and control plane, you will remove workloads, bindings, volumes, export policies, the Astra Data Store cluster, license, deployment environment, and the Astra Data Store namespace.

Uninstalling can be done using different methods:

- [Uninstall Astra Data Store with an automated script](#)
- [Uninstall Astra Data Store manually without a script](#)
- [Troubleshoot the Astra Data Store uninstall process](#)

Uninstall Astra Data Store with an automated script

This process uses an automated script to uninstall Astra Data Store.

What you'll need

- Root administrative permissions

About this task

The Astra Data Store uninstallation process guides you through the following high-level steps:

- [Remove existing workloads and bindings](#)
- [Uninstall Astra Data Store cluster](#)
- [Validate the removal of the astrads-system namespace](#)
- [Ensure containers are not running on worker nodes](#)
- [Delete OpenShift Container Platform resources](#)

Remove existing workloads and bindings

Before uninstalling Astra Data Store, you must first remove the following

- All application workloads that use Astra Data Store as the storage backend
- Trident bindings that use Astra Data Store as a backend

This ensures that your Kubernetes environment is left in a clean state, which is important if you reinstall.

Uninstall Astra Data Store cluster

To uninstall Astra Data Store, you can use the `uninstall.sh` script in your Astra Data Store tar file that was downloaded from the NetApp Support Site.

1. Locate the `uninstall.sh` in the `manifests` directory.
2. Run the following `sed` command:

```
sed -i -e 's~netappsoperator.yaml~astradsoperator.yaml~' uninstall.sh
```

3. Run the following script indicating what you want to uninstall:

```
./uninstall.sh
```

```
You must run this script with an argument specifying what should be
uninstalled
```

```
To uninstall the ADS cluster run ./uninstall.sh cluster
```

```
To uninstall everything run ./uninstall all
```

4. If you want to uninstall just the cluster, enter `uninstall.sh <cluster>`

Otherwise, if you want to uninstall everything, enter `uninstall.sh`



In most cases you will uninstall everything. You might want to uninstall just the cluster if you wanted to redeploy the cluster subsequently.

5. At the prompt, confirm that you want to continue and enter `erasedata`

Response:

```
./uninstall.sh all
```

```
Enter 'erasedata' to confirm you want proceed with the uninstall:
```

```
erasedata
```

```
+-----+
```

```
| Wed Feb  2 10:14:01 EST 2022 |
```

```
| ADS cluster uninstall started |
```

```

+-----+
Deleting astradsvolumes
Deleted astradsvolumes
Deleting astradsexportpolicies
Deleted astradsexportpolicies
Deleting astradsvolumesnapshots
Deleted astradsvolumesnapshots
Deleting astradsclusters
Deleting astradsclusters
Deleting astradslicenses
Deleted astradslicenses

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS cluster uninstall done   |
+-----+

+-----+
| Wed Feb  2 10:15:18 EST 2022 |
| ADS system uninstall started  |
+-----+

Removing astradsversion
astradsversion.astrads.netapp.io "astradsversion" deleted
Removed astradsversion
Removing daemonsets
daemonset.apps "astrads-ds-nodeinfo-astradsversion" deleted
Removed daemonsets
Removing deployments
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted
Removed deployments
Removing all other AstraDS resources
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscLOUDsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted

```



```

customresourcedefinition.apiextensions.k8s.io
"astradsnfsoptions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodeinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodemagements.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsgospolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsversions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumefiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumesnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-admin-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-reader-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-writer-role"
deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
role.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-admin-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-reader-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-writer-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
viewer-role" deleted

```

```
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-  
editor-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-  
viewer-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-editor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-editor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-  
editor-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-  
viewer-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsqospolicy-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-editor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumeeditor-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-viewer-  
role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-  
editor-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-  
viewer-role" deleted  
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted  
rolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-  
rolebinding" deleted  
rolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-  
rolebinding" deleted  
rolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-  
rolebinding" deleted  
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
```

```

rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-
rolebinding" deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
secret "astrads-autosupport-certs" deleted
+-----+
| Wed Feb  2 10:16:36 EST 2022 |
| ADS system uninstall done   |
+-----+

```

Validate the removal of the astrads-system namespace

Ensure that the following command returns no result:

```
kubectl get ns | grep astrads-system
```

Ensure containers are not running on worker nodes

Validate that containers such as `firetap` or `netwd` are not running on the worker nodes. Run the following on each node.

```
ssh <mynode1>
# runc list
```

Delete OpenShift Container Platform resources

If you installed Astra Data Store on Red Hat OpenShift Container Platform (OCP), you can uninstall OCP security context constraints (SCC) and rolebindings resources.

OpenShift uses security context constraints (SCC) that control the actions that a pod can perform.

After you complete the standard uninstall process, complete these steps.

1. Remove SCC resources:

```
oc delete -f ads_privileged_scc.yaml
```

2. Remove rolebindings resources:

```
oc delete -f oc_role_bindings.yaml
```



Ignore "resources not found" errors in these steps.

Uninstall Astra Data Store manually without a script

This process uninstalls Astra Data Store manually without a script.

To uninstall Astra Data Store manually without an automated script, you will remove workloads, bindings, volumes, export policies, clusters, license, deployment environment, and the Astra Data Store namespace.

What you'll need

- Root administrative permissions

About this task

The Astra Data Store uninstallation process guides you through the following high-level steps:

- [Remove existing workloads and bindings](#)
- [Uninstall the Astra Data Store cluster and control plane](#)
- [Delete the license](#)
- [Delete the Astra Data Store installation](#)
- [Validate the removal of the astrads-system namespace](#)
- [Ensure containers are not running on worker nodes](#)
- [Delete OpenShift Container Platform resources](#)

Remove existing workloads and bindings

Before uninstalling Astra Data Store, you must first remove the following

- All application workloads that use Astra Data Store as the storage backend
- Trident bindings that use Astra Data Store as a backend

This ensures that your Kubernetes environment is left in a clean state, which is important if you reinstall.

Uninstall the Astra Data Store cluster and control plane

Follow the steps below to uninstall Astra Data Store manually.

Delete the volumes and export policies

Before deleting the cluster, you should delete the Astra Data Store volumes and export policy.



If you do not first delete volumes and export policies, the cluster deletion process pauses until the Astra Data Store volumes objects are deleted. It is more efficient to remove those items before starting to delete the cluster.

Steps

1. Delete the volumes:

```
~% kubectl delete astradsvolumes --all -A
~% kubectl get astradsvolumes -A
```

2. Delete the export policies:

```
~% kubectl delete astradsexportpolicies --all -A
~% kubectl get astradsexportpolicies -A
```

Delete the Astra Data Store cluster

Deleting the cluster deletes only the Astra Data Store cluster object custom resource (CR) along with cluster-scoped resources.



The operator, nodeinfo pods, and the cluster controller (which are Kubernetes-scoped resources) remain even after the cluster is deleted.

Deleting the cluster also uninstalls the underlying operating system from the nodes, which will stop the `firetap` and `netwd` services.

The uninstaller takes about a minute to finish. Then, the removal of the Astra Data Store cluster-scoped resources starts.

1. Delete the cluster:

```
~% kubectl delete astradsclusters --all -A
~% kubectl get astradsclusters -A
```

Delete the license

1. ssh to each worker node in the cluster and validate that `firetap` or `netwd` are not running in the worker nodes.
2. Delete the Astra Data Store license:

```
~% kubectl delete astradslicenses --all -A
~% kubectl get astradslicenses -A
```

Delete the Astra Data Store installation

Delete the controllers, operators, namespace, and support pods in the cluster.

1. Delete the Astra Data Store installation object:

```
~% kubectl delete astradsversion astradsversion -n astrads-system
~% kubectl get astradsversion -n astrads-system
```

2. Delete the data store DaemonSets and all Astra Data Store controller resources:

```
~% kubectl delete ds --all -n astrads-system
~% kubectl get ds -n astrads-system

~% kubectl delete deployments --all -n astrads-system
~% kubectl get deployments -n astrads-system
```

3. Delete remaining artifacts and the operator yaml file:

```
~% kubectl delete -f ./manifests/astradsoperator.yaml
~% kubectl get pods -n astrads-system
```

Validate the removal of the astrads-system namespace

Ensure that the following command returns no result:

```
~% kubectl get ns | grep astrads-system
```

Ensure containers are not running on worker nodes

Validate that containers such as `firetap` or `netwd` are not running on the worker nodes. Run the following on each node.

```
ssh <mynode1>
# runc list
```

Delete OpenShift Container Platform resources

If you installed Astra Data Store on Red Hat OpenShift Container Platform (OCP), you can uninstall OCP security context constraints (SCC) and rolebindings resources.

OpenShift uses security context constraints (SCC) that control the actions that a pod can perform.

After you complete the standard uninstall process, complete these steps.

1. Remove SCC resources:

```
oc delete -f ads_privileged_scc.yaml
```

2. Remove rolebindings resources:

```
oc delete -f oc_role_bindings.yaml
```



Ignore "resources not found errors" in these steps.

Manual deletion sample

The following shows a sample of an execution manual uninstallation script.

```
$ kubectl delete astradsvolumes --all -A
No resources found
$ kubectl delete astradsexportpolicies --all -A
No resources found
$ kubectl delete astradsclusters --all -A
astradscluster.astrads.netapp.io "astrads-sti-c6220-09-10-11-12" deleted

$ kubectl delete astradslicenses --all -A
astradslicense.astrads.netapp.io "e900000005" deleted

$ kubectl delete astradsdeployment astradsdeployment -n astrads-system
astradsdeployment.astrads.netapp.io "astradsdeployment" deleted

$ kubectl delete ds --all -n astrads-system
daemonset.apps "astrads-ds-astrads-sti-c6220-09-10-11-12" deleted
daemonset.apps "astrads-ds-nodeinfo-astradsdeployment" deleted
daemonset.apps "astrads-ds-support" deleted

$ kubectl delete deployments --all -n astrads-system
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-deployment-support" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted

$ kubectl delete -f ../../firetap/sds/manifests/netappsdsoperator.yaml
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscloudsnapshots.astrads.netapp.io" deleted
```

```
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsdeployments.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsoptions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnodeinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqospolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumefiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvolumesnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-editor-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-autosupport-viewer-role"
deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-metrics-reader" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-editor-
```



```

role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappexportpolicy-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsdeployment-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-netappsdsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-proxy-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-proxy-rolebinding"
deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-fluent-bit-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
service "astrads-operator-metrics-service" deleted
Error from server (NotFound): error when deleting
"/.../export/firetap/sds/manifests/netappsdsoperator.yaml":
deployments.apps "astrads-operator" not found

$ kubectl get ns | grep astrads-system

[root@sti-rx2540-535c ~]# runc list
ID          PID          STATUS      BUNDLE      CREATED      OWNER

```

Troubleshoot the Astra Data Store uninstall process

If you need to troubleshoot the uninstall process, review the following suggestions.

Pods in terminating state

The Astra Data Store uninstall process can occasionally cause pods to remain in a terminating state in Kubernetes.

If this issue occurs, run the following command to force delete all pods in the `astrads-system` namespace:

```
kubectl delete pods --all -n astrads-system --force --grace-period 0
```

Quality of Service policy points to old cluster

If you delete the Astra Data Store Cluster only and redeploy it, you might not be able to create a persistent volume claim (PVC) or volume because the Quality of Service (QoS) policy points to the old cluster and cannot be found.

1. To avoid this, after deleting the Astra Data Store cluster, manually delete the QoS policy:

```
kubectl delete AstraDSQosPolicy --all -A
```

2. Delete the entire Astra Data Store deployment (not just the cluster):

```
uninstall.sh all
```

Key provider CRs not removed after deleting or uninstalling Astra Data Store

If an external key provider has been configured for an Astra Data Store cluster that is being deleted or uninstalled, you might need to manually clean up any key provider CR that was not removed.

Details

Use the following workaround instructions:

Steps

1. Confirm that the key provider CRs were not removed:

```
kubectl get astradskeyprovider --selector  
astrads.netapp.io/cluster=astrads-cluster-example -n astrads-system
```

Response:

| NAME | AGE |
|----------------------|-----|
| externalkeyprovider1 | 94s |

2. Remove the key provider CRs:

- a. Remove the finalizer:

```
kubectl edit astradskeyprovider -n astrads-system
```

- b. Remove the finalizer line highlighted below:

```
kubectl edit astradskeyprovider externalkeyprovider1 -n astrads-  
system
```

```

apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSKeyProvider
metadata:
  creationTimestamp: "2022-05-24T16:38:27Z"
  finalizers:
    - astrads.netapp.io/astradskeyprovider-finalizer
  generation: 1
  labels:
    astrads.netapp.io/cluster: astrads-cluster-example
    astrads.netapp.io/rsid: "1"
  name: externalkeyprovider1
  namespace: astrads-system
  resourceVersion: "1134699"
  uid: a11111b2-31c0-4575-b7f3-97f9ab1a1bla
spec:
  cluster: astrads-cluster-example
  kmipServer:
    hostnames:
      - 10.xxx.xxx.xxx
    port: 5696
    secretRef: externalkeyprovider1
status:
  keyProviderUUID: a1b2cd34-4fc6-5bae-9184-2288c673181d
  kmipServerStatus:
    capabilities: '{ KMIP_library_version()=17367809,
KMIP_library_version_str()="KMIP
1.9.3a 8-Apr-2019", KMIP_library_version_tag()="KMIP part
of KMIP 1.9.3a 8-Apr-2019",
KMIP_library_is_eval()=false,
KMIP_library_fips_capable()=true(FIPS140),
KMIP_SSL_provider_build_version()=268444095,
KMIP_SSL_provider_version()=268444095,
KMIP_SSL_provider_version_str()="OpenSSL
1.0.2zb-fips 23 Sep 2021" }'
  keyServerUUID: 8422bdd0-74ad-579d-81bd-6d544ac4224a

```

c. After the finalizer has been removed, delete the key provider CR:

```

kubectl delete astradskeyprovider <key-provider-cr-name> -n
astrads-system

```

Unable to uninstall Astra Data Store from Astra Control Center web UI

The Astra Data Store uninstallation process can occasionally fail if you initiated it from the Astra Control Center web UI.

If this issue occurs, take the following steps.

Steps

1. Log in to the [NetApp Support Site](#) and download the Astra Data Store bundle (Astra_Data_Store_2022.05.01.tar) to a machine that can access the Kubernetes cluster where Astra Data Store resides.
2. Log in to the machine to which you downloaded the Astra Data Store bundle.
3. Extract the contents of the bundle:

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

4. Change to the manifests directory, where the uninstall script is stored:

```
cd astrads/manifests/
```

5. Manually remove Astra Data Store:

```
./uninstall all
```

Use Astra Data Store with VMware

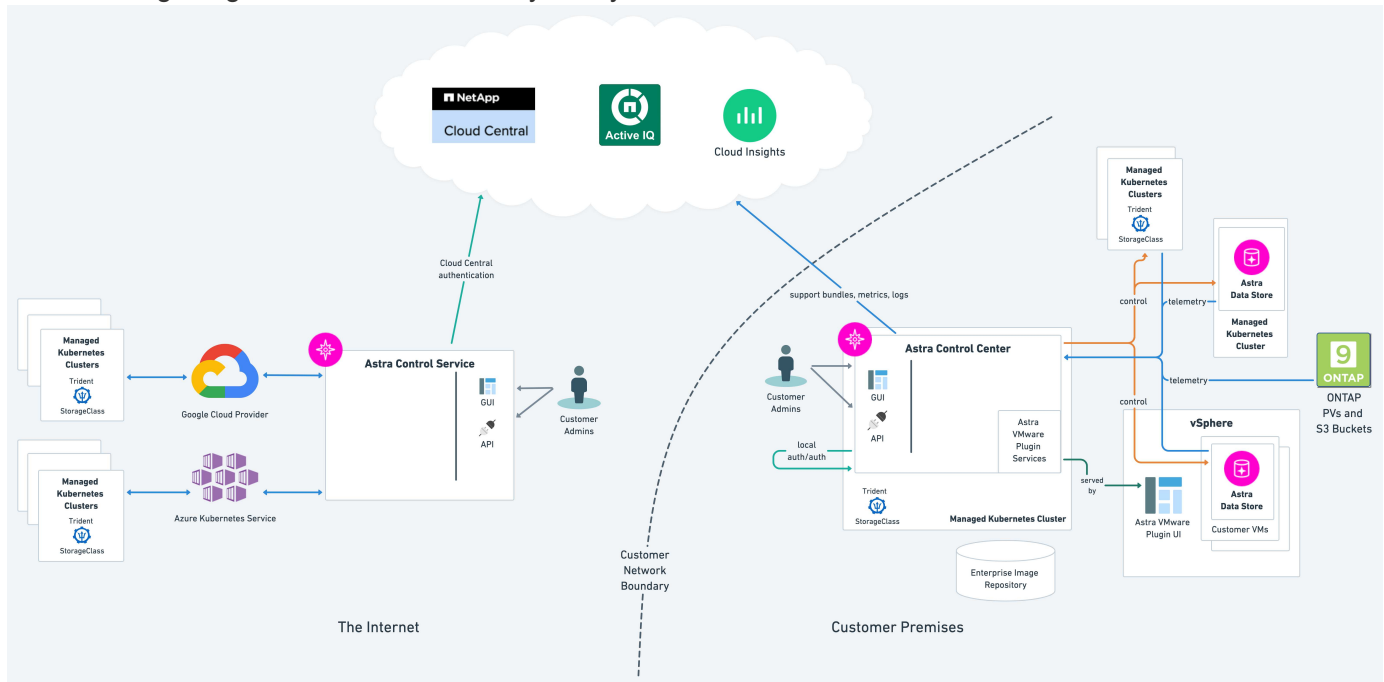
Learn about Astra Data Store with VMware

Astra Data Store supports both containerized and virtualized workloads. Integrations with vVols and storage policy-based management enable vSphere administrators to apply a storage quality of service. The NetApp Astra Plugin for VMware vSphere provides a familiar management and monitoring experience, eliminating cumbersome storage tasks.

The Astra Plugin for VMware vSphere offers the following benefits:

- VM granular storage provisioning with full vVols and VASA integrations
- Storage policy-based management integration
- vCenter plugin for vSphere native management

The following image shows the Astra family ecosystem with VMware.



For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)

Astra Data Store with VMware requirements

After you verify that your environment meets general [Astra Data Store requirements](#), you should verify that your environment also meets minimum requirements for VMware components such as the VMware VASA provider and the NetApp Astra Plugin for VMware vSphere.

VMWare vSphere requirements

Astra Data Store uses the VMWare VASA Provider as an API interface to communicate with storage. Make sure your environment meets the basic [VASA Provider requirements](#) as well as the following additional requirements:

- VMware vSphere 7.0 (Update 1 through Update 3 supported)
- One unassigned IP address for ingress traffic



- NetApp Astra Plugin for VMware vSphere does not support vSphere 7.0 Update 3c; use vSphere 7.0 Update 3d instead.
- NetApp Astra Plugin for VMware vSphere does not support Linked Mode vCenter servers.

NetApp Astra Plugin for VMware vSphere requirements

The NetApp Astra Plugin for VMware vSphere has the following requirements:

- An Astra Control Center instance running on a Kubernetes cluster
- A licensed Astra Data Store instance running on a Kubernetes cluster

Supported web browsers

The NetApp Astra Plugin for VMware vSphere supports the latest versions of the following web browsers:

- Mozilla Firefox
- Microsoft Edge (Chromium-based)
- Google Chrome

For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)
- [Astra Data Store requirements](#)

Set up Astra Data Store with VMware

You can set up Astra Data Store as a storage backend and manage it using the NetApp Astra Plugin for VMware vSphere.

Setting up Astra Data Store with VMware involves the following tasks:

- [Activate VMware vSphere services using Astra Control Center.](#)
- [Add a vCenter using Astra Control Center.](#)
- [Create a custom SCC \(if using OpenShift\)](#)
- [Use an existing storage backend in the Astra Plugin for VMware vSphere.](#)
- [Create a datastore using the Astra Plugin for VMware vSphere.](#)
- [Generate VM storage policies.](#)

What you'll need

Before you set up Astra Data Store with VMware, you'll need to ensure the following:

- Astra Control Center is [installed](#) and set up.



For the Astra Data Store Early Access Program (EAP) release, deploy Astra Control Center only on the `pcloud` namespace if you intend to manage Astra Data Store using Astra Control Center and enable VMware workflows.

- Astra Data Store is licensed and deployed. See [Install Astra Data Store](#).
- The Kubernetes clusters that were used to deploy Astra Control Center and Astra Data Store must already be managed by Astra Control Center.
- You have uploaded your Astra Control Center and VASA provider packages before adding a vCenter. See [Manage software packages](#).

Activate VMware vSphere services using Astra Control Center

Begin setting up Astra Data Store with VMware by activating vSphere services on Astra Control Center.



VMware vSphere services in Astra Control Center are not enabled by default.

1. Log in to Astra Control Center.
2. From the left navigation, select **Clusters**.

The banner displays a message that VMware vSphere services are not yet enabled.

3. Select **Enable VMware vSphere services**.

This action might take a while. After the services are enabled, the **Add vCenter** button is enabled.

Add a vCenter using Astra Control Center

Add your first vCenter, which registers the Astra Plugin for VMware vSphere.

You must have administrative privileges to add the vCenter to Astra Control Center.



After the plugin is registered with VMware vSphere, the Astra Plugin for VMware vSphere icon appears in the VMware Shortcuts page. Sometimes after you register the Astra Plugin for VMware vSphere, the plugin doesn't appear immediately. In this case, wait a few seconds and refresh the browser.

1. Log in to Astra Control Center.
2. From the left navigation, select **Clusters**.
3. Select **Add vCenter**.
4. Enter the vCenter server details, vCenter port, and administrative user name and password to provide them to Astra Control Center.



This enables deployment of the Astra Plugin for this vCenter in the VMware vSphere client.

5. Select **Add**.

The vCenter appears on the Clusters page and the total number of managed vCenters is updated on the Astra Control Dashboard. This also initiates the Astra Plugin for VMware vSphere deployment.

Verify the vCenter addition

The newly added vCenter appears on the Clusters page and on the Dashboard.



Both vCenters and Kubernetes clusters appear on the Astra Control Center Dashboard.

1. Access Astra Control Center.
2. From the left navigation, select **Clusters**.
3. Verify that the newly managed vCenter appears on the Clusters page.
4. From the left navigation, select **Dashboard**.
5. From the Astra Control Center Dashboard, note the new managed vCenter cluster as part of the **Managed** count.



The Managed Clusters count includes both vCenters and Kubernetes clusters.

6. To view cluster details, click the **Managed** count.

The Clusters page appears.

Create a custom SCC (if using OpenShift)

If you are using OpenShift, you can optionally assign security context constraints (SCC) that control the actions that a pod can perform and control what the pod can access.

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 VASA provider pods. Use this procedure to provide the required higher privileges (listed in the sample) to the service accounts used by VASA provider deployments.

Assign a custom SCC to various default service accounts for the Astra Data Store 'ntv-system' namespace, which is a hybrid of privileged and node-exporter SCCs.

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

1. Create a custom SCC called `vp_backend_privileged_scc.yaml`:

```
kubectl create -f vp_backend_privileged_scc.yaml
```

Sample: `vp_backend_privileged_scc.yaml`

```
allowHostDirVolumePlugin: true
allowHostIPC: false
allowHostNetwork: true
```

```

allowHostPID: false
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
  - '*'
allowedUnsafeSysctls:
  - '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  name: vpbbackend-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
  - '*'
supplementalGroups:
  type: RunAsAny
users:
  - system:serviceaccount:ntv-system:default
  - system:serviceaccount:ntv-system:ntv-auth-svc
  - system:serviceaccount:ntv-system:ntv-autosupport
  - system:serviceaccount:ntv-system:ntv-compliance-svc
  - system:serviceaccount:ntv-system:ntv-datastore-svc
  - system:serviceaccount:ntv-system:ntv-metallb-controller
  - system:serviceaccount:ntv-system:ntv-metallb-speaker
  - system:serviceaccount:ntv-system:ntv-mongodb
  - system:serviceaccount:ntv-system:ntv-nfs-svc
  - system:serviceaccount:ntv-system:ntv-rabbitmq-svc
  - system:serviceaccount:ntv-system:ntv-storage-svc
  - system:serviceaccount:ntv-system:ntv-vault
  - system:serviceaccount:ntv-system:ntv-vault-admin
  - system:serviceaccount:ntv-system:ntv-vault-agent-injector
  - system:serviceaccount:ntv-system:ntv-vault-controller
  - system:serviceaccount:ntv-system:ntv-vault-initializer
  - system:serviceaccount:ntv-system:ntv-vcenter-svc
  - system:serviceaccount:ntv-system:ntv-vm-management-svc

```

```

- system:serviceaccount:ntv-system:ntv-watcher-svc
- system:serviceaccount:ntv-system:ntv-vault-sa-vault-tls
- system:serviceaccount:ntv-system:ntv-gateway-svc
- system:serviceaccount:ntv-system:ntv-jobmanager-svc
- system:serviceaccount:ntv-system:ntv-vasa-svc
volumes:
- '*'

```

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

```
oc get scc vpbakend-privileged
```

Response:

| NAME | PRIV | CAPS | SELINUX | RUNASUSER | FSGROUP | SUPGROUP |
|---------------------|-------|-------|----------|-----------|----------|----------|
| vpbakend-privileged | true | ["*"] | RunAsAny | RunAsAny | RunAsAny | RunAsAny |
| <no value> | false | ["*"] | | | | |

Use an existing storage backend in the Astra Plugin for VMware vSphere

After adding a vCenter by using the Astra Control Center UI, add the Astra Data Store storage backend using the Astra Plugin for VMware vSphere.

This process completes the following actions:

- Adds an existing storage backend to the selected vCenter.
- Registers the VASA provider with the selected vCenter. The VASA provider provides communication between VMware and Astra Data Store.
- Adds a VASA provider self-signed certificate to the storage backend.



It can sometimes take a few minutes for the storage backend you added to appear in the storage backend wizard.



Astra Data Store should not be shared with multiple vCenters.

Steps

1. Access the NetApp Astra Plugin for VMware vSphere.
2. From the left navigation, select **Astra Plugin for VMware vSphere** or from the Shortcuts page, select the **Astra Plugin for VMware vSphere** icon.
3. From the Astra Plugin for VMware vSphere Overview page, select **Use existing storage backend**. Or, from the left navigation, select **Storage Backends > Add**, and select **Use existing storage backend**.
4. Select the existing Astra Data Store as the storage backend and select **Next**.

5. On the VASA provider page, enter the VASA provider name, IP address (if using a load balancer), user name, and password.



For the user name, you can use alphanumeric characters and the underscore. Do not enter any special characters. The first letter of the user name must begin with an alphabet character.

6. Indicate whether you want to deploy a load balancer and enter the IP address, which will be used to access the VASA provider. The IP needs to be an additional routable free IP separate from the node IPs. When the load balancer is enabled, Metallb is deployed in the Astra Data Store Kubernetes cluster and configured to allocate the free IP.



If you are using a Google Anthos cluster for deployment, choose not to deploy a load balancer as Anthos already runs metallb as a load balancer. The metallb deploy flag should be set to false in VASA provider CR (v1beta1_vasaprovider.yaml).

If you choose not to deploy a load balancer, it is assumed that the load balancer has already been deployed and configured to allocate IPs for the Kubernetes service of type **Load Balancer**.



At this point in the deployment, the VASA provider is not yet deployed.

7. Select **Next**.
8. On the Certificate page, review the certificate information for the self-signed certificate.
9. Select **Next**.
10. Review summary information.
11. Select **Add**.

This deploys the VASA provider.

Verify the storage backend in the Astra Plugin for VMware vSphere

After the Astra Data Store storage backend is registered, it appears in the Astra Plugin for VMware vSphere storage backends list.

You can determine the storage backend status and the VASA provider status. You can also see the used capacity of each storage backend.

After selecting a storage backend, you can also view used and available capacity, data reduction ratio, and internal network management IP address.

Steps

1. In the NetApp Astra Plugin for VMware vSphere, from the left navigation, select **Storage Backends**.
2. Select the Astra Data Store storage backend to see the Summary tab.
3. Review used and available capacity, data reduction ratio, and status of the VASA provider.
4. Select the other tabs to see information about VMs, datastores, hosts, and storage nodes.

Create a datastore using the Astra Plugin for VMware vSphere

After adding the storage backend and registering the Astra Plugin for VMware vSphere, you can create a

datastore in VMware.

You can add the datastore to a datacenter, compute, or a host cluster.



You cannot use the same storage backend to create multiple datastores under same datacenter.

You can add a vVol datastore type using an NFS protocol.

Steps

1. Access the Astra Plugin for VMware vSphere.
2. From the plugin menu, select **Create Datastore**.
3. Enter the new datastore name, type (vVol), and protocol (NFS).
4. Select **Next**.
5. From the Storage page, select the Astra Data Store storage backend that you just created.



You cannot use a storage backend that has an existing datastore.

6. Select **Next**.
7. From the Summary page, review the information.
8. Select **Create**.



If you encounter an error related to a failed scan or general system error, [rescan/synchronize your storage provider on vCenter](#) then try to create the datastore again.

Generate VM storage policies

After you create a datastore and before you create VMs, you should generate predesigned VM storage policies by using `/virtualization/api/v1/vcenters/vm-storage-policies` in the REST API UI.

Steps

1. Access the REST API UI page by going to https://<ads_gateway_ip>:8443.
2. Go to the API POST `/virtualization/api/auth/login` and provide the username, password and vCenter hostname.

Response:

```
{
  "vmware-api-session-id": "212f4d6447b05586ab1509a76c6e7da56d29cc5b",
  "vcenter-guid": "8e475060-b3c8-4267-bf0f-9d472d592d39"
}
```

3. Go to the API GET `/virtualization/api/auth/validate-session` and complete the following steps:
 - a. Use the `vmware-api-session-id` and `vcenter-guid` generated above as headers.
 - b. Select **Try it now**.

Response: (authentication truncated below):

```
authorization: eyJhbGciOiJSUzI1NiIsInR...9h15DYYvClT3oA  connection:
keep-alive  content-type: application/json  date: Wed, 18 May 2022
13:31:18 GMT  server: nginx  transfer-encoding: chunked
```

4. Go to the API `/virtualization/api/v1/vcenters/vm-storage-policies` and add the bearer token generated in the previous response as 'authorization'.

A "200" response appears and three VM storage policies are generated.

5. Verify the new VM storage policies (named Bronze, Silver, and Gold) on the VCenter Storage Policy page.
6. Continue by creating VMs.

What's next

Next, you might want to do the following tasks:

- Create VMs.
- Mount the datastore. See [Mount a datastore](#).

For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)

Monitor components of your VMware installation

You can use the NetApp Astra Plugin for VMware vSphere to monitor components of your Astra Data Store installation. You can monitor the health of your system including the storage backends, VASA providers, VMs, and vVols. You can also view capacity and vCenter information.

Monitor the health of your system using the Astra Plugin for VMware vSphere Dashboard

Managing your Astra Data Store with VMware environment involves monitoring the overall health of storage backends and VASA providers.

Using the NetApp Astra Plugin for VMware vSphere Dashboard, you can see the following information:

- Physical used and available capacity of all storage backends in this vCenter. You can hover over information and see more details.
- Healthy and unhealthy storage backends and VASA providers
- Latency, IOPS, throughput and capacity utilization for the top 10 VMs.

From the Dashboard, you can perform several additional tasks:

- Monitor capacity

- Use an existing storage backend. See [Set up storage backends](#).
- Access the product documentation

Steps to review the Dashboard

1. Access the Astra Plugin for VMware vSphere.
2. From the Overview page, review the following sections:
 - a. **Storage backends** section: You can click on the states of both the storage backends and VASA providers to see details about their status. You can also click to view all storage backends.
 - b. **Storage Backend Capacity** section: Review the total physical used and available capacity for the storage backends in the selected vCenter. To change the vCenter server, click the vCenter Server option in the upper right.
 - c. **Virtual Machines** section: Review virtual machines having the top 10 capacity utilization.



You can instead show what you want, for example, the top 10 VMs with high latency, by clicking on the table heading.

Steps to monitor Astra Data Store in other views

1. Access the following views to monitor Astra Data Store components:
 - **Virtual Machines** tab: Lists all the VMs that are managed by Astra Data Store compared to the Dashboard, which lists only the top 10 VMs.
 - **Storage** drill down: Displays hosts, virtual machines and datastores that are associated with the storage system.
 - **VM Storage** view: Provides details of vVols that are created by the VASA provider.

Review storage backend threshold settings

Storage backend capacity threshold settings govern when alert notifications appear on all datastores in the storage backend.

The following default thresholds are set when you deploy or add a storage backend by using the Astra Plugin for VMware vSphere:

- 90% full generates a red alert
- 80% full generates a yellow alert

You can view the level at which the system generates an alert in VMware.



For the Astra Data Store Early Access Program, if the same storage container is used across multiple datacenters, you might see an incorrect alarm for datastores.

Steps

1. Access the NetApp Astra Plugin for VMware vSphere.
2. From the left navigation, select **Settings**.
3. Review the settings thresholds.

For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)

Manage Astra Data Store components of your VMware installation

You can manage the following Astra Data Store components in your vSphere environment and from Astra Control Center:

- [Work with managed vCenters](#)
- [Manage VMs from vSphere](#)
- [Manage the storage backend](#)
- [Manage datastores](#)

Work with managed vCenters

You can work with the managed vCenters in the following ways:

- [View vCenter details in Astra Control Center](#)
- [View vCenter details in Astra Control Center](#)
- [Unmanage a vCenter in Astra Control Center](#)

View vCenter details in Astra Control Center

You can determine all the vCenters associated with a cluster.

Steps

1. From the Astra Control Center left navigation, select **Clusters**.
2. View the list of vCenters.
3. Select **Save**.

View vCenter details in Astra Control Center

You might want to view the health of your system and clusters. You can determine how many clusters are managed by looking at the Astra Control Center Dashboard.

Steps

1. From the Astra Control Center left navigation, select **Clusters**.
2. Select the vCenter.
3. View the information.

Unmanage a vCenter in Astra Control Center

If you no longer want Astra Control Center to manage the vCenter, you can unmanage it. This removes and unregisters the vCenter from Astra Control Center.



Clusters, storage backends and VMs associated with this vCenter must be removed from the Astra Plugin for VMware vSphere first.

Steps

1. From the Astra Control Center left navigation, select **Clusters**.
2. From the Clusters page, select the vCenter.



Or, select multiple vCenters and select **Unmanage all**.

3. Right-click the **Actions** menu and select **Unmanage**.
4. In the Unmanage vCenter page, type "unmanage".
5. Select **Yes, unmanage vCenter**.

Manage VMs from vSphere

You can manage VMs associated with Astra Data Store using native vSphere operations.

- [Delete VM](#)
- [Rename VM](#)
- [Resize VM](#)



For this release, you can only resize one VM disk at a time. Attempts to resize multiple disks will fail.

- [Power VM on or off](#)
- [Suspend VM](#)
- [Reset VM](#)

The following snapshot workflows are available for the Astra Data Store using native vCenter operations:

- [Take snapshot of Astra Data Store](#)
- [Revert a snapshot](#)
- [Delete a snapshot](#)



Snapshot operations might occasionally fail with a vVol runtime error. If this happens, retry the operation.

Manage the storage backend

You can remove the storage backend. Removing a storage backend does not destroy it and does not delete the Astra Data Store product itself; it just unregisters the VASA provider from VMware and unlinks the storage backend for that vCenter.



If the VASA provider is enabled and deployed outside of the vCenter, you can only remove Astra Data Store. If the storage backend is being used as part of a provisioning datastore process, you won't be able to remove the storage backend.

If Astra Data Store is not being linked with more than one vCenter, when you remove it, the VASA provider will be unregistered and uninstalled.

Steps

1. Access the Astra Plugin for VMware vSphere.
2. From the left navigation, select **Storage Backends**.
3. On the Storage Backends page, click on the storage backend Actions menu and select **Remove**.
4. Enter the VASA provider user name and password.
5. Select **Remove**.

Manage datastores

You can manage Astra Data Store in your vSphere environment using native vCenter operations to manage VMs and Astra Plugin extensions to manage datastores:

- [Create a datastore](#)
- [Mount a datastore](#)
- [Delete a datastore](#)

Mount a datastore

Using the Astra Plugin for VMware vSphere, you can mount the datastore on one or more additional hosts.

Steps

1. Select the datastore for Astra Data Store from your data center inventory in vCenter.
2. Right-click the datastore and select **Astra Plugin for VMware vSphere > Mount Datastore**.
3. From the Mount datastore on hosts page, select the hosts on which you want to mount the datastore.



If you want to mount the datastore on all hosts, check **Mount on all hosts**.

4. Select **Mount**.

After you initiate the operation, you can follow progress in the Recent Tasks panel in the vSphere Client.



If you encounter an error related to a failed scan or general system error, [rescan/synchronize your storage provider on vCenter](#) then try to create the datastore again.

Delete a datastore

Using the Astra Plugin for VMware vSphere, you can delete a datastore.



To delete the datastore, all VMs on the datastore must first be removed.

Steps

1. Select the datastore from your data center inventory in vCenter.
2. Right-click the datastore and select **Astra Plugin > Delete Datastore**.
3. In the Delete Datastore page, confirm the information or take additional suggested actions so that the datastore can be deleted.

4. Select **Delete**.

For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)

Uninstall Astra Data Store from a VMware-integrated environment

You can uninstall Astra Data Store and its related components from your vSphere environment.

See [these instructions](#) on uninstalling Astra Data Store.

For more information

- [Astra Control Center documentation](#)
- [Astra family introduction](#)

Knowledge and support

Troubleshooting

Learn how to work around some common problems you might encounter.

https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra

Get help

NetApp provides support for Astra Data Store in a variety of ways. Free self-service support options are available 24x7, such as knowledgebase (KB) articles and a Slack channel.



You can get community technical support for Astra Data Store. Case creation using the [NetApp Support Site \(NSS\)](#) is not available for this release. You can get in touch with Support via the feedback option or use the Slack channel for self service.

Self-service support options

These options are available for free 24x7:

- [Knowledge base \(login required\)](#)

Search for articles, FAQs, or Break Fix information related to Astra Data Store.

- Documentation

This is the doc site that you're currently viewing.

- [NetApp "containers" Slack channel](#)

Go to the "containers" channel to connect with peers and experts.

- Feedback email

Send an email to astra.feedback@netapp.com to let us know your thoughts, ideas, or concerns.

Find more information

- [How to upload a file to NetApp \(login required\)](#)
- [NetApp Knowledge Base articles](#)

Automatic support monitoring

AutoSupport monitors the Astra Data Store system run-time and information and sends messages to NetApp Support. These system components can be monitored, depending on your configuration:

- Control plane
- Storage

AutoSupport is enabled by default during [Astra Data Store cluster installation](#) or after an AutoSupport custom resource (CR) is applied to the cluster. Once enabled, AutoSupport (ASUP) bundles are automatically uploaded to the [NetApp Support Site \(NSS\)](#) or made available for manual downloads.

Options

- [AutoSupport triggers and scenarios](#)
- [Configure custom control plane AutoSupport collection](#)
- [Configure custom storage AutoSupport collection](#)
- [List ASUPs in the system](#)
- [Download an ASUP Bundle](#)
- [Upload a core file](#)

AutoSupport triggers and scenarios

AutoSupport bundles are triggered in the following ways:

- **Periodically:** ASUP Bundles are created at intervals defined in a CR.
- **User Triggered:** You can manually create your own ASUPs to look at the log.
- **Coredumps:** If there are core dumps on a node, an ASUP is generated, and the core is sent to NetApp for further investigation.
- **Callhome event based:** An ASUP is generated from a particular callhome event from the operating system.
- **Kubernetes event based:** An ASUP is generated from a particular kubernetes event in the control plane.

These trigger scenarios generate one of these Autosupport types:

- **ControlPlane AutoSupport:** A collection of Astra Data Store control plane logs and CRs.
- **Storage AutoSupport:** A collection of storage reports and performance data.
- **Core Dump AutoSupport:** A collection of system core dumps.

Configure custom control plane AutoSupport collection

You can create a custom AutoSupport collection configuration that reports on control plane events. Most installations already enable periodic event reporting by default during [Astra Data Store cluster installation](#). This procedure describes how to configure an AutoSupport CR that reports based on parameters you select:

Steps

1. Customize the following command to create a control plane collection CR:

```
kubectl astasds asup collect -c controlplane --namespace=astrads-system
```

- a. Define custom parameters:

- `<myASUPname>`: The name of the AutoSupport CR to be generated.
- `-e <event name>`: The event name that triggers collection. The event name should be predefined in component.yaml (which is mounted to support controllers).

Example:

```
kubectl astrasds asup collect -c controlplane custom-asup-name -e
debug --namespace=astrads-system
```

b. Add additional parameters as needed for your system:

- `--cluster`: This flag is required in a multi-cluster environment.
- `--localCollection`: Enables local collection. The default is `false`.
- `--forceUpload`: Enables force upload. The default is `false`.
- `--retry`: Enables retry. The default is `false`.

Configure custom storage AutoSupport collection

You can create a custom AutoSupport collection configuration that reports on storage component events. Most installations already enable periodic event reporting by default during [Astra Data Store cluster installation](#). This procedure describes how to configure an AutoSupport CR that reports based on parameters you select:

Steps

1. Customize the following command to create a storage collection CR:

```
kubectl astrasds asup collect -c storage --namespace=astrads-system
```

a. Define custom parameters:

- `<myASUPname>`: The name of the AutoSupport CR to be generated.
- `-e <event name>`: The event name that triggers collection. The event name should be predefined in `component.yaml` (which is mounted to support controllers).

Example with performance event:

```
kubectl astrasds asup collect -c storage -e performance example-
perf-storage-asup
```

- `-t <ISO_format> -d <hours>`: Collect a storage ASUP for all nodes for a specified duration. Use standard ISO date time format (`-t`) with a duration (`d`) in hours. For example:

```
kubectl astrasds asup collect -c storage -t 2021-01-01T15:00:00Z -d
24
```

- `--nodes <nodename>`: Collect a storage ASUP for specified node. For example:

```
kubectl astrasds asup collect -c storage --nodes example1
```

- `--nodes nodename1,nodename2,nodename3`: Collect a storage ASUP for specified nodes:

```
kubectl astrads asup collect -c storage --nodes
example1,example2,example3
```

b. Add additional parameters as needed for your system:

- `--cluster`: This flag is required in a multi-cluster environment.
- `--localCollection`: Enables local collection. The default is `false`.
- `--forceUpload`: Enables force upload. The default is `false`.
- `--retry`: Enables retry. The default is `false`.

List ASUPs in the system

Use the following command to list ASUPs in the system by name:

```
kubectl astrads asup list --namespace=astrads-system
```

Sample response:

| NAMESPACE | NAME | SEQUENCE | NUMBER | EVENT |
|----------------|-------------------------------------|----------|------------|---------------------------|
| SIZE | STATE | LOCAL | COLLECTION | |
| astrads-system | storage-callhome.reboot.unknown-... | 1 | | |
| | callhome.reboot.unknown | 0 | uploaded | astrads-ds-support-tdl2h: |
| astrads-system | storage-callhome.reboot.unknown-... | 2 | | |
| | callhome.reboot.unknown | 0 | uploaded | astrads-ds-support-xx6n8: |
| astrads-system | storage-callhome.reboot.unknown-... | 3 | | |
| | callhome.reboot.unknown | 0 | uploaded | astrads-ds-support-qghnx: |

Download an ASUP Bundle

You can download locally-collected ASUP bundles using this command. Use `-o <location>` to specify a location other than the current working directory:

```
./kubectl-astrads asup download <ASUP_bundle_name> -o <location>
```

Upload a core file

If a service crashes, an AutoSupport (ASUP) message is created along with a file containing relevant memory contents at the time of the crash (known as a core file). Astra Data Store automatically uploads the ASUP message to NetApp Support, but you need to manually upload the core file so that it is associated with the ASUP message.

Steps

1. Use the following `kubectl` commands to view the ASUP message:

```
kubectl astrads asup list --namespace=astrads-system
```

You should see output similar to the following:

| NAMESPACE | NAME | SEQUENCE | NUMBER | EVENT |
|----------------|--------------------------|-----------------------------------|------------|----------|
| SIZE | STATE | LOCAL | COLLECTION | |
| astrads-system | storage-coredump-2021... | 1 | | coredump |
| 197848373 | compressed | astrads-ds-support-sxxn7:/var/... | | |

2. Use the following `kubectl` commands to download the core file from the ASUP message. Use the `-o` option to specify a destination directory for the downloaded file.

```
kubectl astrads asup download storage-coredump-20211216t140851311961680  
-o <absolute_path_to_destination_directory>
```



In rare cases, you might not be able to download the core file because other core files have taken its place. When this happens, the command returns the error `Cannot stat: No such file or directory`. If you see this error, you can [get help](#).

3. Open a web browser and browse to the [NetApp Authenticated File Upload tool](#), entering your NetApp Support credentials if you are not already logged in.
4. Select the **I don't have a case number** check box.
5. In the **Closest Region** menu, select the closest region to you.
6. Select the **Upload** button.
7. Browse to and select the core file you downloaded earlier.

The upload begins. When the upload is finished, a success message appears.

Find more information

- [How to upload a file to NetApp \(login required\)](#)

Earlier versions of Astra Data Store

Documentation for previous releases is available.

- [Astra Data Store 21.12 documentation](#)

Legal notices

Legal notices provide access to copyright statements, trademarks, patents, and more.

Copyright

<http://www.netapp.com/us/legal/copyright.aspx>

Trademarks

NETAPP, the NETAPP logo, and the marks listed on the NetApp Trademarks page are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

<http://www.netapp.com/us/legal/netapptmlist.aspx>

Patents

A current list of NetApp owned patents can be found at:

<https://www.netapp.com/us/media/patents-page.pdf>

Privacy policy

<https://www.netapp.com/us/legal/privacypolicy/index.aspx>

Open source

Notice files provide information about third-party copyright and licenses used in NetApp software.

[Notice for Astra Data Store EAP 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.