NetApp

Astra Data Store preview documentation

Astra Data Store

NetApp
February 16, 2022

Table of Contents

Astra Data Store preview documentation

Release notes

We're pleased to announce the 21.12 preview release of Astra Data Store preview.

- What's new in this release of Astra Data Store preview
- 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 2021.12.1 patch of the preview release of Astra Data Store.

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 preview release of the product successfully.

MongoDB deployment with default liveness probe value fails with pods in crash loop

As a workaround, set the liveness probe initialDelaySeconds to 600 seconds in the MongoDB deployment spec.

Astra Data Store preview uninstallation process might cause pods to remain in a terminating state

The Astra Data Store preview uninstall process in Kubernetes v1.20 can occasionally cause pods to remain in a terminating state.

See Uninstall Astra Data Store preview manually without a script or Uninstall Astra Data Store preview automatically with a script.

Find more information

- Known limitations

Known limitations

Known limitations identify platforms, devices, or functions that are not supported by this preview release of the product, or that do not interoperate correctly with it. Review these limitations carefully.

Functionality to remove one or more nodes is not supported

Astra Data Store preview supports replacing a failed node but does not support node removal functionality.

Functionality to add or remove drives is not supported

Astra Data Store preview supports replacing a failed drive but does not support adding or removing drives to or from an existing cluster.

Astra Data Store preview functionality is not validated with firewall enabled

Astra Data Store preview requires host firewalld to be disabled. Enabling firewall using CNI tools like Calico HostEndpoint has not been validated.

Upgrade or patch requires a fresh install

Astra Data Store preview is not targeted for production workloads.

Ubuntu-based bare-metal or VM passthrough deployments require NVMe TLC SSDs

This limitation is not applicable to RHEL, RHCOS, or CentOS-based deployments.

Find more information

- Known issues

Concepts

Intro to Astra Data Store preview

Astra Data Store preview 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 preview, 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 product family

The Astra family of products provides Kubernetes application data lifecycle management that simplifies operations for stateful applications. Easily protect, back up, and migrate Kubernetes workloads, and instantly create working application clones.

Astra offerings include:

- **Astra Control**: Use application-aware, data management tools that manage, protect, and move Kubernetes workloads in both public clouds and on-premises.
  - **Astra Control Service**: Use a service managed by NetApp for data management of Kubernetes workloads in public clouds.
  - **Astra Control Center**: Use self-managed software for data management of on-premises Kubernetes workloads.
- **Astra Data Store preview**: Use a Kubernetes-native shared file service for container and VM workloads for enterprise data management.
- **Astra Trident**: Use Container Storage Interface (CSI) compliant storage provisioning and management for Kubernetes workloads with NetApp storage providers.

See Astra family introduction.

Astra Data Store preview features

Astra Data Store preview 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 preview

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 preview deployment models

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

You can install Astra Data Store preview 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 the Astra Data Store hardware requirements.

Astra Data Store preview is part of the Astra product family. To get a perspective on the entire Astra family, see Astra family introduction.

Astra Data Store preview ecosystem

Astra Data Store preview 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 OpenShift Kubernetes clusters with an Astra Trident storage backend with ONTAP or an Astra Data Store preview 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 preview.

- **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 preview.

Astra Data Store preview 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 preview 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 preview with the Astra Control API.
- **kubectl commands**: To work with Astra Data Store preview, you can use kubectl commands directly.
- **Kubernetes extension**: Additionally, you can use the Astra Data Store preview kubernetes API extension.

  Custom Resource Definitions (CRDs) are extensions to the kubernetes REST API that are created when the Astra Data Store preview operator is deployed. External entities interact with the CRDs by making calls to the Kubernetes API server. Astra Data Store preview 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

Cluster expansion

Astra Data Store preview supports nodes of different types and capabilities in a cluster. If you are expanding a cluster, Astra Data Store preview 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 preview 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 preview

Astra Data Store preview 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**: ONTAP 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 ONTAP 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 preview

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

Astra Data Store preview 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 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 preview requirements

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

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

Astra Data Store preview supports provisioning only of persistent volumes for Kubernetes workloads using the Astra Trident CSI driver. VM workloads will be supported in a future release of Astra Data Store.

> ⓘ If you plan to manage your Astra Data Store preview cluster from Astra Control Center, make sure your Astra Data Store preview cluster meets the requirements for clusters that will be managed by Astra Control Center in addition to the requirements outlined here.

Kubernetes worker node resource requirements

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

Resource	Minimum	Maximum
Number of data drives	• 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 or greater in size)	N/A
Number of vCPUs	10	10
RAM	35GiB	35GiB

> ⓘ 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 preview 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 requirements

Astra Data Store preview 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

- HPE DL360
- HPE DL380
- Dell R640
- Dell R740

Storage

Astra Data Store preview has been validated with the following drive types:

- **Bare-metal deployments**: Astra Data Store preview installed on a Kubernetes cluster directly on a Linux cluster without any hypervisor
    - NVMe TLC SSDs
- **VM-based deployments**: Astra Data Store preview installed on a Kubernetes cluster on Linux VMs hosted on an ESXi cluster
    - SAS or NVMe TLC SSD-based datastores
    - 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 preview is validated with VMware-based VM deployments with ESXi 7.0. KVM-based deployments are not supported by Astra Data Store preview.
- Astra Data Store preview 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 preview has been validated with the following Kubernetes distributions:
    - Red Hat OpenShift 4.7
    - Google Anthos 1.7
    - Kubernetes 1.21
    - Kubernetes 1.20

> ⓘ   Astra Data Store preview requires Astra Trident version 21.10.1 for storage provisioning and orchestration. See the Astra Trident installation instructions.

Networking requirements

Astra Data Store preview 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 preview 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:

> ℹ️ 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	• Two (2) per node:
	   ◦ 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	• Three per node:
	   ◦ 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

> ℹ️ You should omit the data network gateway field in the cluster Custom Resource (CR) file, `astradscluster.yaml`, for both of these configurations. The existing routing configuration on each node accommodates all of the addresses.

> ℹ️ No VLAN tags are used in these configurations.

Astra Trident

Astra Data Store preview requires the application Kubernetes clusters to be running Astra Trident 21.10.1. Astra Data Store preview can be configured as a storage backend with Astra Trident to provision persistent volumes.

CNI configuration

Astra Data Store preview has been validated with the following CNIs:

- Calico and Weave Net CNIs for vanilla Kubernetes clusters
- OpenShift SDN for Red Hat OpenShift Container Platform (OCP)

- Cilium for Google Anthos

These CNIs require the host firewall (firewalld) to be disabled.

Persistent volume sharing requirements

Each Astra Data Store preview cluster supports using persistent volumes to address the storage needs of any apps installed on that cluster. Consider the following requirements for persistent volumes in Astra Data Store preview:

Requirements

- The NFSv4.1 client/server must be installed on Kubernetes clusters.
- The nfs-utils package must be installed on worker nodes.
- Kubernetes apps access files using persistent volumes shared over NFSv4.1, which requires the AUTH_SYS authentication method.

Licensing

Astra Data Store preview requires an Astra Data Store preview license for full functionality. Sign up here to obtain the Astra Data Store preview license. Instructions to download the license will be sent to you after you sign up.

AutoSupport configuration

Astra Data Store preview requires AutoSupport to be enabled and have connectivity to the AutoSupport backend. This may be through direct internet access or proxy configuration.

The periodic settings that are used for sending mandatory telemetry AutoSupport bundles should not be changed. If you disable the sending of periodic AutoSupport bundles, the cluster will be locked down and new volumes cannot be created until periodic settings are enabled again.

What's next

View the quick start overview.

For more information

Astra Data Store preview limits

Quick start for Astra Data Store preview

This page provides a high-level overview of the steps needed to get started with Astra Data Store preview. 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 preview, you can use a 90-day preview license.

Sign up here to obtain the Astra Data Store preview 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 preview 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 preview cluster.
- Each SSD should have a unique serial number.

Learn more about Astra Data Store preview requirements.

2   Download and install Astra Data Store preview

- Download Astra Data Store preview from the NetApp Support Site.
- Install Astra Data Store preview in your local environment.
- Apply the Astra Data Store preview license.
- Install the Astra Data Store preview cluster.
- Configure Astra Data Store preview monitoring.
- If you use Red Hat OpenShift, install Astra Data Store preview on Red Hat OpenShift Container Platform (OCP).

Learn more about installing Astra Data Store preview.

3   Complete some initial setup tasks

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

Learn more about the initial setup process.

4   Use Astra Data Store preview

After you finish setting up Astra Data Store preview, 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 preview.
- Configure monitoring endpoints. Learn more about configuring monitoring endpoints.

5   Continue from this Quick Start

Install Astra Data Store preview.

Installation overview for Astra Data Store

Choose and complete one of the following Astra Data Store installation procedures:

- Install Astra Data Store using the standard process.

- If you use Red Hat OpenShift, install Astra Data Store using OpenShift.

Install Astra Data Store preview

To install Astra Data Store preview, download the installation bundle from the NetApp Support Site and complete installation steps described in this procedure.

Alternatively, you can install Astra Data Store preview on Red Hat OpenShift Container Platform (OCP).

What you'll need

- Before you begin installation, prepare your environment for Astra Data Store preview deployment.
- Access to the NetApp Support Site. Register for a preview if you don't already have a full-access NetApp Support Site account.
- A NetApp license file (NLF) for Astra Data Store preview. 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 preview.
- Internet connectivity. Astra Data Store preview does not support air-gapped environments. Internet connectivity is needed to reach support.netapp.com either directly or via a proxy.

About this task

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

- Download the Astra Data Store preview bundle and extract the images
- Copy the binary and push images to your local registry
- Install the Astra Data Store preview operator
- Deploy the Astra Data Store preview version YAML

- Apply the Astra Data Store preview license
- Install the Astra Data Store preview cluster
- Understand deployment-related events
- Configure Astra Data Store preview monitoring

If you want to enable Astra Data Store preview to work with image registries with secrets, see this KB.

Download the Astra Data Store preview bundle and extract the images

1. Log in to the NetApp Support Site and download the Astra Data Store preview bundle (`2021.12_astradatastore.tar`).

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

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. Extract the images:

```
tar -xvf 2021.12_astradatastore.tar
```

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; for example, `/usr/bin/`. kubectl-astrads is a custom kubectl extension that installs and manages Astra Data Store preview clusters.

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

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

   (i) | 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 preview 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 $(ls images/*.tar) ; do
    astraImage=$(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}'~' ./manifests/*.yaml
```

Install the Astra Data Store preview operator

1. List the Astra Data Store preview manifests:

```
ls manifests/*yaml
```

Response:

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

2. Deploy the operator using kubectl apply:

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

Response:

```
namespace/astrads-system 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/astradsdeployments.astrads
.netapp.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/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/astradsqospolicies.astrads
.netapp.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-leader-election-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-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-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-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
```

```
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-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 preview version YAML

1. Deploy using kubectl apply:

```
kubectl apply -f ./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      1/1      Running   0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk         1/1      Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v         1/1      Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt         1/1      Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv         1/1      Running   0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j       1/1      Running   0
2m2s
astrads-operator-5ffb94fbf-7ln4h                 1/1      Running   0
2m10s
```

Apply the Astra Data Store preview license

1. Apply the NetApp License File (NLF) that you obtained when you signed up for the preview. 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         ADSCLUSTER                  VALID    PRODUCT
   EVALUATION   ENDDATE       VALIDATED
   p100000006   astrads-example-cluster      true    Astra Data Store Preview
   true         2022-01-23  2021-11-04T14:38:54Z
   ```

Install the Astra Data Store preview cluster

1. Open the YAML file:

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

2. Edit the following values in the YAML file.

> ℹ️ 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 `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 preview nodes. For Astra Data Store preview, this means at least 4 addresses, or 5 if you plan on expanding the cluster to 5 nodes 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.

> ℹ️ Cluster and storage networks need to be on the same interface. The Astra Data Store preview 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.

> ℹ️ Most comments have been removed from the YAML sample below.

```yaml
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
```

```yaml
    cpu: 9
    memory: 34
  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 k8s label key that defines which protection
domain a node belongs to.
    # adsProtectionDomainKey: ""
  # [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 ./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 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-1ab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

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     2021.10.0     p100000006
10.x.x.x     10m
```

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	Meaning
Successfully selected 4 control plane nodes to join the ADS cluster	The Astra Data Store preview operator identified enough nodes with CPU, memory, storage, and networking to create an Astra Data Store preview cluster.
ADS cluster create in progress	The Astra Data Store preview cluster controller has started the cluster create operation.
ADS cluster created	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 at `in progress`, check the cluster controller's logs:

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

Configure Astra Data Store preview monitoring

You can configure Astra Data Store preview for Astra Control Center monitoring or for monitoring by another telemetry service.

Configure monitoring for Astra Control Center preview

Perform the following step only after Astra Data Store preview is managed as a backend in Astra Control Center.

1. Configure Astra Data Store preview for monitoring by Astra Control Center:

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

Install the monitoring operator

(Optional) The monitoring operator is recommended if Astra Data Store preview will not be imported into Astra Control Center. You can install the monitoring operator if your Astra Data Store preview instance is a standalone deployment, uses Cloud Insights to monitor telemetry, or streams logs to a third-party endpoint such as Elastic.

1. Run this install command:

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

2. Configure Astra Data Store preview for monitoring:

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

What's next

Complete the deployment by performing setup tasks.

Install Astra Data Store preview on Red Hat OpenShift Container Platform

To install Astra Data Store preview on Red Hat OpenShift Container Platform (OCP), download the installation bundle from the NetApp Support Site and complete 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 a preview if you don't already have a full-access NetApp Support Site account.
- A NetApp license file (NLF) for Astra Data Store preview. 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 preview.
- Internet connectivity. Astra Data Store Preview does not support air-gapped environments. Internet connectivity is needed to reach support.netapp.com either directly or via a proxy.

About this task

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

- Download the Astra Data Store preview bundle and extract the images
- Copy the binary and push images to your local registry
- Create a namespace to deploy Astra Data Store preview
- Create a custom SCC
- Create the roles and role bindings
- Install the Astra Data Store preview operator
- Deploy the Astra Data Store preview version YAML
- Apply the Astra Data Store preview license
- Install the Astra Data Store preview cluster
- Understand deployment-related events
- Configure Astra Data Store preview monitoring
- Install the monitoring operator

If you want to enable Astra Data Store preview to work with image registries with secrets, see this KB.

Download the Astra Data Store preview bundle and extract the images

1. Log in to the NetApp Support Site and download the Astra Data Store preview bundle
   (`2021.12_astradatastore.tar`).

2. (Optional) Verify the signature of the bundle:

```
openssl dgst -sha256 -verify 2021.12_astradatastore.pub -signature
2021.12_astradatastore.sig 2021.12_astradatastore.tar
```

3. Extract the images:

```
tar -xvf 2021.12_astradatastore.tar
```

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; for example, `/usr/bin/`. kubectl-astrads is a custom kubectl extension
   that installs and manages Astra Data Store preview clusters.

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

2. Add the files in the Astra Data Store preview 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 preview 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 $(ls images/*.tar) ; do
    astraImage=$(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}'~' ./manifests/*.yaml
```

Create a namespace to deploy Astra Data Store preview

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

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 preview cluster pods. Use this procedure to provide the required privileges (listed in the sample) to Astra Data Store preview.

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

Steps

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 preview.

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

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
```

Prepare the worker nodes

Prepare the worker nodes for Astra Data Store preview cluster deployment. Perform this procedure on all the worker nodes used by the Astra Data Store preview cluster.

OpenShift uses a json format for the kubelet config file (`/var/lib/kubelet/config.json`). The Astra Data Store preview cluster looks for the yaml format of the `kubelet config` file.

Steps

1. Create `/var/lib/kubelet/config.yaml` file on each of the worker nodes before initiating the cluster installation.

```
sudo cp /var/lib/kubelet/config.json /var/lib/kubelet/config.yaml
```

2. Complete this procedure on all Kubernetes nodes before the cluster yaml is applied.

> (i) If you do not do this, the Astra Data Store preview cluster installation will fail.

Install the Astra Data Store preview operator

1. List the Astra Data Store preview manifests:

```
ls manifests/*yaml
```

Response:

```
manifests/astradscluster.yaml
manifests/astradsoperator.yaml
manifests/astradsversion.yaml
manifests/monitoring_operator.yaml
```

2. Deploy the operator by using the `kubectl apply` command:

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

Response:

```
namespace/astrads-system 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/astradsdeployments.astrads
.netapp.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/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/astradsqospolicies.astrads
.netapp.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-leader-election-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-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-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-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
```

```
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-proxy-role created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-proxy-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-fluent-bit-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
service/astrads-operator-metrics-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 preview version YAML

1. Deploy by using the `kubectl apply` command:

```
kubectl apply -f ./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       1/1      Running    0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk          1/1      Running    0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v          1/1      Running    0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt          1/1      Running    0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv          1/1      Running    0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j        1/1      Running    0
2m2s
astrads-operator-5ffb94fbf-7ln4h                  1/1      Running    0
2m10s
```

Apply the Astra Data Store preview license

1. Apply the NetApp License File (NLF) that you obtained when you signed up for the preview. 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          ADSCLUSTER                      VALID    PRODUCT
   EVALUATION    ENDDATE       VALIDATED
   p100000006    astrads-example-cluster    true     Astra Data Store Preview
   true          2022-01-23    2021-11-04T14:38:54Z
   ```

Install the Astra Data Store preview cluster

1. Open the YAML file:

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

2. Edit the following values in the YAML file.

> (i) 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 `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 preview nodes. For Astra Data Store preview, this means at least 4 addresses, or 5 if you plan on expanding the cluster to 5 nodes 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.

      > (i) Cluster and storage networks need to be on the same interface. The Astra Data Store preview 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.

      > (i) Most comments have been removed from the YAML sample below.

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
```

```
      cpu: 9
      memory: 34
  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 k8s label key that defines which protection
domain a node belongs to.
    # adsProtectionDomainKey: ""
  # [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 ./manifests/astradscluster.yaml
```

4. If SELinux is enabled, re-label the `selinux` context for the following directories on the nodes in the Astra Data Store preview cluster.

```
sudo chcon -R -t container_file_t
/var/opt/netapp/firetap/rootfs/var/asup/notification/firetap/
```

```
sudo chcon -R -t container_file_t /var/netapp/firetap/firegen/persist/
```

> (i) This step is needed because `selinux` prevents these directories from being writable, causing the support pods to enter a `CrashLoopBackoff` state. This step needs to be performed on all the nodes in the Astra Data Store preview cluster.

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

6. Verify the cluster deployment progress:

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

Sample response:

```
NAME                           STATUS    VERSION    SERIAL NUMBER    MVIP
AGE

astrads-example-cluster   created   2021.10.0   p100000006
10.x.x.x    10m
```

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
```

40

The following are key events during deployment:

Event message	Meaning
Successfully selected 4 control plane nodes to join the ADS cluster	The Astra Data Store preview operator identified enough nodes with CPU, memory, storage, and networking to create an Astra Data Store preview cluster.
ADS cluster create in progress	The Astra Data Store preview cluster controller has started the cluster create operation.
ADS cluster created	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 at `in progress`, check the cluster controller's logs:

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

Configure Astra Data Store preview monitoring

You can configure Astra Data Store preview for Astra Control Center monitoring or for monitoring by another telemetry service.

Configure monitoring for Astra Control Center preview

Perform the following step only after Astra Data Store preview is managed as a backend in Astra Control Center.

1. Configure Astra Data Store preview for monitoring by Astra Control Center:

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

Install the monitoring operator

(Optional) The monitoring operator is recommended if Astra Data Store preview will not be imported into Astra Control Center. You can install the monitoring operator if your Astra Data Store preview instance is a standalone deployment, uses Cloud Insights to monitor telemetry, or streams logs to a third-party endpoint such as Elastic.

1. Run this install command:

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

2. Configure Astra Data Store preview for monitoring:

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

What's next

Complete the deployment by performing setup tasks.

Set up Astra Data Store preview components

After you have installed Astra Data Store preview 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:

- 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

Install Astra Trident

For Astra Data Store preview, you'll need to install Astra Trident 21.10.1. 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 preview 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.snap
shot.storage.k8s.io created
```

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

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

Response:

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotcontents.sna
pshot.storage.k8s.io created
```

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 created
```

4. Apply setup-snapshot-controller.yaml:

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

Response:

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

5. Apply rbac-snapshot-controller.yaml:

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

Response:

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

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. 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": "bronze"
    },
    "storage": [
        {
            "labels": {
                "performance": "extreme"
            },
            "defaults": {
                "qosPolicy": "bronze"
            }
        },
        {
            "labels": {
                "performance": "premium"
            },
            "defaults": {
                "qosPolicy": "bronze"
            }
        },
        {
            "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
   ```

   Response:

   ```
   apiVersion: storage.k8s.io/v1
   kind: StorageClass
   metadata:
     name: trident-csi
   provisioner: csi.trident.netapp.io
   reclaimPolicy: Delete
   volumeBindingMode: Immediate
   allowVolumeExpansion: true
   mountOptions:
     - vers=4.1
   ```

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 -A
```

Response:

```
NAME            PROVISIONER              RECLAIMPOLICY    VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION    AGE
trident-csi    csi.trident.netapp.io    Delete           Immediate
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
```

```yaml
      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: bronze
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
      region: ""
      supportedTopologies: null
      zone: ""
    - defaults:
        exportPolicy: ""
        qosPolicy: bronze
        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:
        - trident-csi
        supportedTopologies: null
  volumes: []
```

Astra Data Store preview limits

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.

The Astra Data Store preview release has the following resource limits.

Resource	Minimum	Maximum
Number of nodes in Astra Data Store preview cluster	4	5
Number of persistent volumes per node	N/A	10

Resource	Minimum	Maximum
Total provisioned capacity of persistent volumes per node	N/A	1TiB
Volume size	20MiB	1TiB
Snapshots per volume	0	256
Clones per Volume	0	9

> ⓘ Astra Data Store preview does not support VM workloads. VMware vVol workload support will be available in a future release.

> ⓘ Astra Data Store preview is performance throttled and should not be used for performance characterization.

Frequently asked questions for Astra Data Store preview

Find answers to the frequently asked questions about installing, configuring, upgrading, and troubleshooting Astra Data Store preview.

General questions

Can I use Astra Data Store preview for production?
No. While Astra Data Store is designed and developed to deliver enterprise grade resiliency, as a preview release Astra Data Store preview is not targeted for production workloads.

Can I use Astra Data Store preview for virtual machine workloads?
Astra Data Store preview release is limited only to applications running on Kubernetes, whether on bare metal or virtual machines. Future releases will support applications on both Kubernetes and directly on ESXi virtual machines. See Astra Data Store requirements.

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

Yes. Astra Data Store preview 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.

Applications using Astra Data Store preview cluster as a storage backend can use Astra Control Center version 21.12 to leverage application-aware data management capabilities, including data protection, disaster recovery, and migration of Kubernetes workloads.

How can I manage Astra Data Store preview cluster?
You can manage Astra Data Store preview assets 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 preview cluster metrics?
You can monitor Astra Data Store preview metrics using Cloud Insights. See Monitor metrics with Cloud Insights.

You can also monitor logs. See Configure and monitor event logs.

Can I use Astra Data Store preview along with ONTAP or other storage providers in my Kubernetes clusters?
Yes. Astra Data Store preview 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 preview?
Astra Trident will not be uninstalled from the cluster if you uninstall Astra Data Store preview. If you require Astra Trident to be uninstalled, you will have to do it separately.

Licensing

Does the Astra Data Store preview require a license?
Yes, Astra Data Store preview requires a NetApp license file (NLF).

See Astra Data Store requirements.

How long is the Astra Data Store preview license valid?
The Astra Data Store preview license has a default term of 90 days from the download date.

Installation and use of Astra Data Store preview on a Kubernetes cluster

Can I install Astra Data Store preview on a Kubernetes cluster running on bare metal or virtual machines?
Yes. Astra Data Store preview can be installed on a Kubernetes cluster running on bare metal or on ESXi VMs. See Astra Data Store preview requirements.

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

Astra Data Store preview 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 preview requirements.

My Kubernetes cluster is larger than 5 worker nodes. Can I install Astra Data Store preview on it?
Yes. An Astra Data Store preview cluster can be deployed on 4 worker nodes in a Kubernetes cluster. After deployment, you can grow the cluster to 5 worker nodes.

Does Astra Data Store preview support an offline installation from a private registry?
Yes. Astra Data Store preview can be installed offline from a local registry. See Install Astra Data Store preview. However, Astra Data Store preview requires connectivity (direct or via a proxy) to the NetApp AutoSupport backend (support.netapp.com) for continued operations.

Do I need internet connectivity to use Astra Data Store preview?
Astra Data Store preview requires connectivity to the NetApp AutoSupport backend to periodically send the mandatory telemetry AutoSupport bundles. This connectivity can be direct or via a proxy. If this connectivity is missing or AutoSupport is disabled, the cluster will lock down and new volume creation will be disabled until the periodic bundle uploads are resumed.

What are the roles and privileges used by Astra Data Store preview?
You need to be a kube admin to deploy the Astra Data Store preview operator.

Astra Data Store preview has a privileged daemonset called `astrads-ds-nodeinfo-astradsversion` 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 preview storage cluster.

What manifest files do I need to update for Astra Data Store preview installation?
From the Astra Data Store preview 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 preview.

Troubleshooting and support

With the Astra Data Store preview, 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

The preview release requires that your system is connected to the cloud and integrated into the NetApp Active IQ and AutoSupport tools.

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 preview.

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

Can IPv6 addresses be used for management, data, and cluster networks?
No, Astra Data Store preview only supports IPv4 address. IPv6 support will be added in a future release of Astra Data Store preview.

What NFS version is used while provisioning a volume on Astra Data Store preview?
By default, Astra Data Store preview supports NFS v4.1 for all volumes provisioned for Kubernetes applications.

Why can't I get larger persistent volumes even though I have configured Astra Data Store preview with large capacity drives?
Astra Data Store preview limits the maximum capacity provisioned for all volumes on a node to 1 TiB and up to 5 TiB across all nodes in an Astra Data Store preview cluster.

See Astra Data Store preview requirements and Astra Data Store preview limits.

Upgrading Astra Data Store preview

Can I upgrade from Astra Data Store preview release?
No. Astra Data Store preview is not for production workloads and new releases of Astra Data Store preview software will require a fresh installation.

Use Astra Data Store

Manage Astra Data Store preview assets with kubectl commands

You can manage Astra Data Store preview assets 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 the following cluster maintenance information, see Manage the cluster:

- Place a node in maintenance mode
- Replace a drive
- Add a node
- Replace a node

What you'll need

- The Astra Data Store preview kubectl plugin you installed in Install Astra Data Store preview

List Kubernetes custom API resources for Astra Data Store preview

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

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

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

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

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

    Response:

```
NAME                        SHORTNAMES   APIGROUP             NAMESPACED   KIND
astradsversions             adsve        astrads.netapp.io    true
AstraDSVersion
astradsclusters             adscl        astrads.netapp.io    true
AstraDSCluster
astradslicenses             adsli        astrads.netapp.io    true
AstraDSLicense
astradsnodeinfoes           adsni        astrads.netapp.io    true
AstraDSNodeInfo
astradsvolumes              adsvo        astrads.netapp.io    true
AstraDSVolume
astradsqospolicies          adsqp        astrads.netapp.io    true
AstraDSQosPolicy
astradsexportpolicies       adsep        astrads.netapp.io    true
AstraDSExportPolicy
astradsvolumesnapshots      adsvs        astrads.netapp.io    true
AstraDSVolumeSnapshot
astradsvolumefiles          adsvf        astrads.netapp.io    true
AstraDSVolumeFiles
astradsautosupports         adsas        astrads.netapp.io    true
AstraDSAutoSupport
astradsfaileddrives         adsfd        astrads.netapp.io    true
AstraDSFailedDrive
astradsnodemanagements      adsnm        astrads.netapp.io    true
AstraDSNodeManagement
```

2. To get all the current Astra Data Store preview 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                                  ADSCLUSTER
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

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

1. Show help for all commands in the Astra Data Store preview 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:
      --as string                        Username to impersonate for the
operation
```

```
      --as-group stringArray              Group to impersonate for the
operation, this flag can be
                                          repeated to specify multiple
groups.
      --cache-dir string                  Default HTTP cache directory
                                           (default "/u/arda/.kube/http-
cache")
      --certificate-authority string  Path to a cert file for the
certificate authority
      --client-certificate string     Path to a client certificate file
for TLS
      --client-key string             Path to a client key file for TLS
      --cluster string                The name of the kubeconfig
cluster to use
      --context string                The name of the kubeconfig
context to use
  -h, --help                          help for astrads
      --insecure-skip-tls-verify        If true, the server's
certificate will not be checked
                                          for validity. This will make
your HTTPS connections insecure
      --kubeconfig string               Path to the kubeconfig file to
use for CLI requests.
  -n, --namespace string              If present, the namespace scope
for this CLI request
      --request-timeout string          The length of time to wait
before giving up on 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")
  -s, --server string                 The address and port of the
Kubernetes API server
      --token string                  Bearer token for authentication
to the API server
      --user 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.
                                   This should be a positive integer
      -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
                                   automatic uploads of AutoSupports is
disabled
                                   at the cluster level.
      -h, --help                 help for collect
      -l, --local                Only collect and compress the
autosupport bundle. Do not upload
```

```
                                                to support.
                                                Use 'download' to copy the collected
bundle after it is in
                                                the 'compressed' state
            --nodes string              Specify nodes to collect for storage
component. (default "all")
      -t, --startTime string     StartTime is the starting time for
collection of AutoSupport.

                                                This should be in the ISO 8601 date
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.
                                                (default "Manual event trigger from
CLI")
```

Deploy a test application

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

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

What you'll need
- Astra Data Store preview 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 cluster

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

- Add a node
- Place a node in maintenance mode
- Replace a node
- Replace a drive

What you'll need

- System with kubectl and kubectl-astrads plugin installed. See Install Astra Data Store preview.

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.

Steps

1. If the new node's dataIP is not already part of the ADSCluster CR, do the following:

    a. Edit the astradscluster CR and add the additional dataIP in the ADS Data Networks Addresses field:

    ```
    ~% kubectl edit astradscluster <cluster-name> -n astrads-system
    ```

    Response:

    ```
    adsDataNetworks:
        -addresses:  dataIP1,dataIP2,dataIP3,dataIP4,*newdataIP*
    ```

    b. Save the CR.

    c. Add the node to the Astra Data Store preview cluster:

    ```
    ~% kubectl astrads nodes add –cluster <cluster-name>
    ```

2. Otherwise, just add the nodes:

    ```
    ~% kubectl astrads nodes add –cluster <cluster-name>
    ```

3. Verify that the node has been added:

    ```
    ~% kubectl astrads nodes list
    ```

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 preview 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 mainteance mode.

Steps

1. Display the node details:

    ```
    ~% kubectl get nodes
    ```

    Response:

```
NAME                STATUS    ROLES                    AGE      VERSION
  nhcitjj1525       Ready     <none>                   3d18h    v1.20.0
  nhcitjj1526       Ready     <none>                   3d18h    v1.20.0
  nhcitjj1527       Ready     <none>                   3d18h    v1.20.0
  nhcitjj1528       Ready     <none>                   3d18h    v1.20.0
  scs000039783-1    Ready     control-plane,master     3d18h    v1.20.0
```

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.

```
~% kubectl astrads maintenance create <cr-name> --node-name=<<node
-name>> --variant=Node
```

Sample:

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

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
```

Replace a node

Use kubectl commands with Astra Data Store preview to replace a failed 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. Describe the cluster:

```
~% kubectl astrads clusters list
```

Response:

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

3. Verify that `Node HA` is marked as `false` on the failed node:

```
~% kubectl describe astradscluster -n astrads-system
```

Response:

```
Name:        cluster-multinodes-21209
Namespace:   astrads-system
Labels:      <none>
Annotations: kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"astrads.netapp.io/v1alpha1","kind":"AstraDSCluster","meta
data":{"annotations":{},"name":"cluster-multinodes-21209","namespa...
API Version:  astrads.netapp.io/v1alpha1
Kind:        AstraDSCluster

State:              Disabled
Variant:            None
Node HA:            false
Node ID:            4
Node Is Reachable:  false
Node Management IP:  172.21.192.192
Node Name:           sti-rx2540-532d.ctl.gdl.englab.netapp.com
Node Role:           Storage
Node UUID:           6f6b88f3-8411-56e5-b1f0-a8e8d0c946db
Node Version:        12.75.0.6167444
Status:              Added
```

4. Modify the astradscluster CR to remove the failed node by decrementing the value of `AdsNode Count' to 3:

```
cat manifests/astradscluster.yaml
```

Response:

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
```

```
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
spec:
  # ADS Node Configuration per node settings
  adsNodeConfig:
    # Specify CPU limit for ADS components
    # Supported value: 9
    cpu: 9
    # Specify Memory Limit in GiB for ADS Components.
    # Your kubernetes worker nodes need to have at least this much RAM
free
    # for ADS to function correctly
    # Supported value: 34
    memory: 34
    # [Optional] Specify raw storage consumption limit. The operator
will only select drives for a node up to this limit
    capacity: 600
    # [Optional] Set a cache device if you do not want auto detection
e.g. /dev/sdb
    # cacheDevice: ""
    # Set this regex filter to select drives for ADS cluster
    # drivesFilter: ".*"

  # [Optional] Specify node selector labels to select the nodes for
creating ADS cluster
  # adsNodeSelector:
  #   matchLabels:
  #     customLabelKey: customLabelValue

  # Specify the number of nodes that should be used for creating ADS
cluster
  adsNodeCount: 3

  # Specify the IP address of a floating management IP routable from any
worker node in the cluster
  mvip: "172..."

  # Comma separated list of floating IP addresses routable from any host
where you intend to mount a NetApp Volume
  # at least one per node must be specified
  # addresses: 10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.5
  # netmask: 255.255.255.0
  adsDataNetworks:
    - addresses: "172..."
      netmask: 255.255.252.0
```

```
    # [Optional] Provide a k8s label key that defines which protection
domain a node belongs to
    # adsProtectionDomainKey: ""

    # [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
    monitoringConfig:
     namespace: "netapp-monitoring"
     repo: "docker.repo.eng.netapp.com/global/astra"


    autoSupportConfig:
       # AutoUpload defines the flag to enable or disable AutoSupport
upload in the cluster (true/false)
       autoUpload: true
       # Enabled defines the flag to enable or disable automatic
AutoSupport collection.
       # When set to false, periodic and event driven AutoSupport
collection would be disabled.
       # It is still possible to trigger an AutoSupport manually while
AutoSupport is disabled
       # enabled: true
       # CoredumpUpload defines the flag to enable or disable the upload of
coredumps for this ADS Cluster
       # coredumpUpload: false
       # HistoryRetentionCount defines the number of local (not uploaded)
AutoSupport Custom Resources to retain in the cluster before deletion
       historyRetentionCount: 25
       # DestinationURL defines the endpoint to transfer the AutoSupport
bundle collection
       destinationURL: "https://testbed.netapp.com/put/AsupPut"
       # ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
       # proxyURL:
       # Periodic defines the config for periodic/scheduled AutoSupport
objects
       periodic:
         # Schedule defines the Kubernetes Cronjob schedule
         - schedule: "0 0 * * *"
           # PeriodicConfig defines the fields needed to create the
Periodic AutoSupports
           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
```

5. Verify the node is removed from the cluster:

```
~% kubectl get nodes --show-labels
```

Response:

```
NAME                     STATUS   ROLES                 AGE    VERSION
LABELS
sti-astramaster-237    Ready control-plane,master   24h    v1.20.0
sti-rx2540-532d          Ready   <none>                24h    v1.20.0
sti-rx2540-533d          Ready   <none>                24h
```

```
~% 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
sti-rx2540-536d    Added            cluster-multinodes-21209
```

```
~% kubectl get nodes --show-labels
```

Response:

```
NAME                     STATUS   ROLES                 AGE    VERSION
LABELS
sti-astramaster-237 Ready    control-plane,master   24h    v1.20.0
beta.kubernetes.io/arch=amd64,
sti-rx2540-532d      Ready    <none>                24h    v1.20.0
astrads.netapp.io/node-removal
```

```
~% kubectl describe astradscluster -n astrads-system
```

Response:

```
Name:        cluster-multinodes-21209
Namespace:   astrads-system
Labels:      <none>
Kind:        AstraDSCluster
Metadata:
...
```

6. Add a node to the cluster for replacement by modifying the cluster CR. The node count increments to 4. Verify that new node is picked up for addition.

```
rvi manifests/astradscluster.yaml
cat manifests/astradscluster.yaml
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
```

```
~% kubectl apply -f manifests/astradscluster.yaml
```

Response:

```
astradscluster.astrads.netapp.io/cluster-multinodes-21209 configured
```

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

Response:

```
NAME                                  READY   STATUS    RESTARTS   AGE
astrads-cluster-controller...         1/1     Running   1          24h
astrads-deployment-support...         3/3     Running   0          24h
astrads-ds-cluster-multinodes-21209 1/1     Running
```

```
~% 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
```

```
~% kubectl astrads clusters list
```

Response:

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

```
~% kubectl astrads drives list
```

Response:

```
DRIVE NAME      DRIVE ID      DRIVE STATUS    NODE NAME       CLUSTER NAME
scsi-36000..  c3e197f2... Active              sti-rx2540... cluster-
multinodes-21209
```

Replace a drive

When a drive fails in a cluster, the drive must be replaced as soon as possible to ensure data integrity.
When a drive fails, you will see failed drive information in cluster CR node status, cluster health condition
information, and the metrics endpoint.

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 automtically in the cluster with a name corresponding to the UUID of the failed drive.

```
$ kubectl 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.ctl.gdl.englab.netapp.com
   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 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
Drive replacement completed successfully
```

> (i) 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 replacedrive replace --cluster astrads-cluster-
f51b10a --name 6000c2927 --replaceWith VALID_SERIAL_NUMBER --force
```

For more information

- Manage Astra Data Store preview assets with kubectl commands

Monitor metrics with Cloud Insights

You can monitor Astra Data Store preview metrics using Cloud Insights.

- Complete Cloud Insights connection prerequisite tasks

- Acquisition Unit storage

- Download and run the installation script

- Edit the Cloud Insights connection

- Disconnect from Cloud Insights

Here are some sample Astra Data Store preview metrics displayed in Cloud Insights:
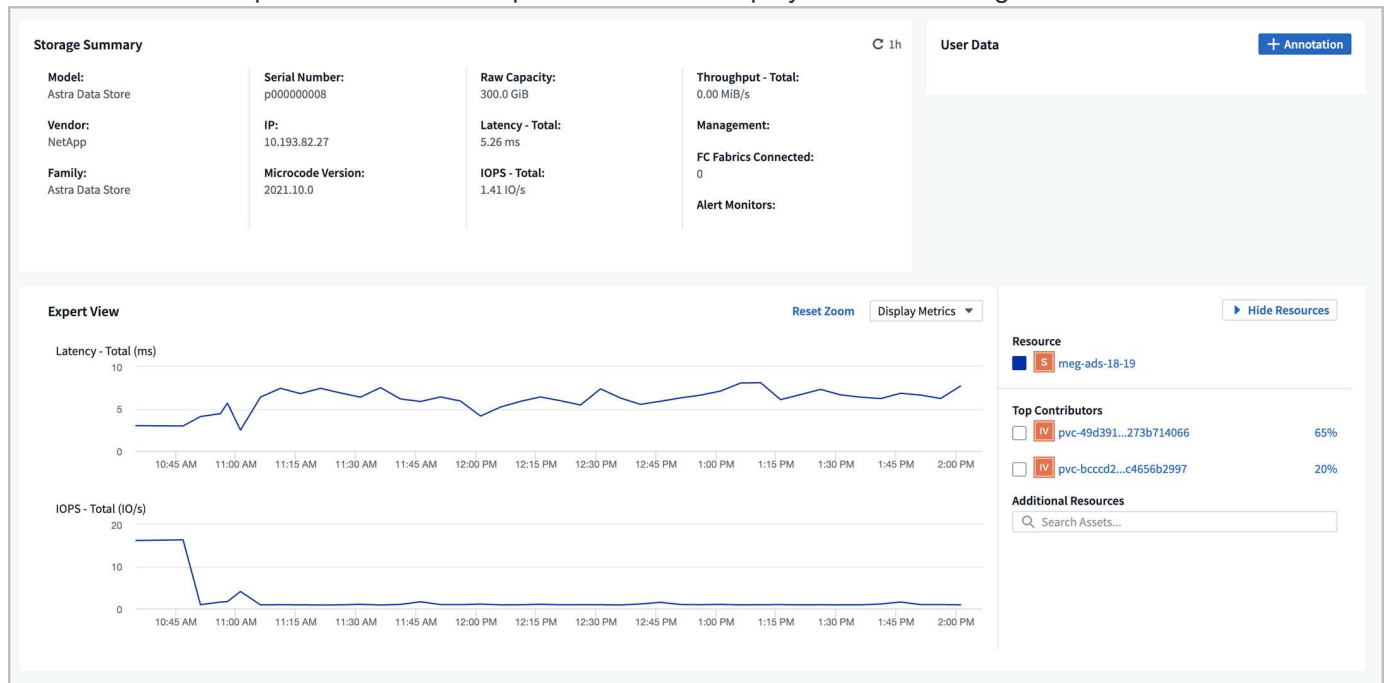
You can also display a list of metrics generated in Astra Data Store preview by using the Open Metrics API help.

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 preview installation instructions.

- Install the kubectl-astrads binary that is part of the Astra Data Store preview 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 preview 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 preview monitoring via the Monitoring Operator. The install script will install an Acquisition Unit with the Astra Data Store collector, a Telegraf agent, 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:

- Telegraf 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.
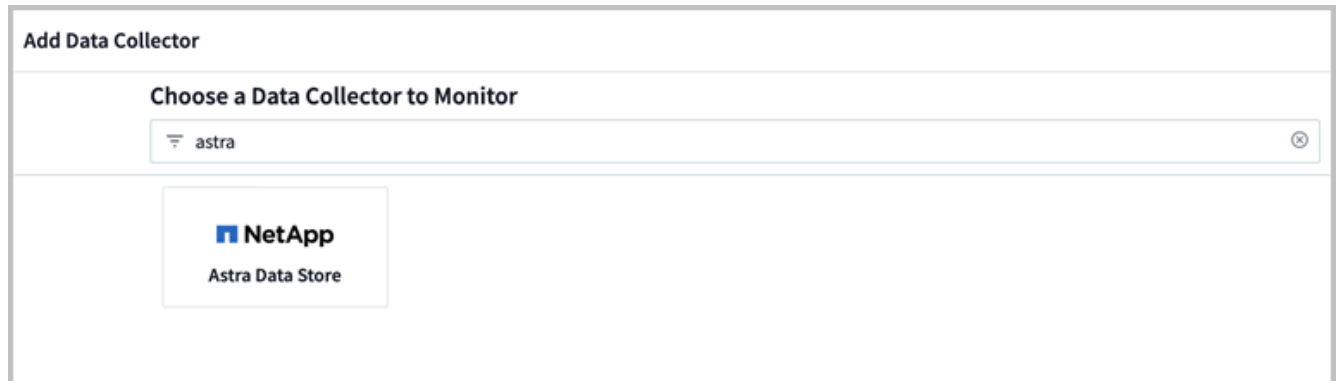
5. Click on the **Astra Data Store** tile.

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.

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

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

> ℹ️ 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 Telegraf 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: bzl9ngz.gst-adsdemo.ci-dev.netapp.com
    name: CI
  serviceAccount: sa-netapp-monitoring
  telegraf:
  - name: ads-open-metric
    outputs:
    - sink: CI
    run-mode:
    - ReplicaSet
    substitutions:
    - key: URLS
      values:
      - http://astrads-metrics-service.astrads-
system.svc.cluster.local:9341
    - key: METRIC_TYPE
      value: ads-metric
    - key: ADS_CATEGORY
      value: netapp_ads
    - key: ADS_CLUSTER_NAME
      value: meg-ads-21-22-28-29-30
  - name: agent
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 Telegraf and 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 and Fluent Bit configurations from the Monitoring Operator.

Remove the Astra Data Store preview 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 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. 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 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 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 preview.

- 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 thoughput 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 thoughput for a volume
in bytes
# TYPE astrads_volume_write_throughput counter
```

Monitor metrics with Prometheus and Grafana

You can monitor Astra Data Store preview metrics with Prometheus and Grafana. You can configure
Prometheus to gather metrics from the Astra Data Store preview 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 preview cluster or a different cluster that can communicate with the Astra Data Store preview
  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 preview Kubernetes
  cluster. If Prometheus and Grafana are not installed on the Astra Data Store preview cluster, you need to
  make sure they can communicate with the metrics service running on the Astra Data Store preview cluster.

Configure Prometheus

Astra Data Store preview 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 preview 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.yaml` 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 preview 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 preview and how you can view them.

Steps

1. Open the Astra Data Store preview `.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 preview cluster custom resource (CR)
- Set up Cloud Insights
- Stream event logs to Elastic.

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

If the monitoring option has not been configured on the Astra Data Store preview 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 preview.

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.

Use Astra Control Center with Astra Data Store preview

You can use the Astra Control Center user interface (UI) to perform Astra Data Store preview tasks.

Set up Astra Control Center for Astra Data Store preview

To use the Astra Control Center UI for Astra Data Store preview, you need to complete these tasks:

- Add the underlying Kubernetes cluster running Astra Data Store to Astra Control Center.
- Add Astra Data Store preview as a storage backend to Astra Control Center.

  > If you add a storage backend and no Kubernetes clusters with Astra Data Store preview exist, you'll need to first add a cluster.

What you can do in Astra Control Center

After you set up Astra Control Center for Astra Data Store preview, you can then use the Astra Control Center UI to accomplish these tasks:

- Monitor the health of your Astra Data Store preview assets using Astra Control Center.
- Manage the Astra Data Store preview backend storage.
- Monitor nodes, disks, and persistent volume claims (PVCs).

For more information

- Astra family introduction
- Astra Control Center documentation
- Astra Control API

Uninstall Astra Data Store preview with an automated script

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

Alternatively, you can uninstall Astra Data Store preview without a script.

What you'll need

- Root administrative permissions

About this task

The Astra Data Store preview 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
- Troubleshoot the Astra Data Store preview uninstall process

Remove existing workloads and bindings

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

- All application workloads that use Astra Data Store preview as the storage backend
- Trident bindings that use Astra Data Store preview 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 preview, 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~netappsdsoperator.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
"astradsnodemanagements.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqospolicies.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-astradsvolumefile-editor-
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 preview 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
```

> ℹ️ Ignore "resources not found" errors in these steps.

3. Remove `/var/lib/kubelet/config.yaml` from all Kubernetes nodes.

Troubleshoot the Astra Data Store preview uninstall process

The Astra Data Store preview uninstall process in Kubernetes v1.20 can occasionally cause pods to remain in a terminating state.

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
```

Uninstall Astra Data Store preview without a script

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

Alternatively, you can uninstall Astra Data Store preview with a script.

What you'll need

- Root administrative permissions

About this task

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

- Remove existing workloads and bindings
- Uninstall the Astra Data Store preview cluster and control plane
- Delete the license
- Delete the Astra Data Store preview installation
- Validate the removal of the astrads-system namespace
- Ensure containers are not running on worker nodes
- Delete OpenShift Container Platform resources
- Troubleshoot the Astra Data Store preview uninstall process

Remove existing workloads and bindings

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

- All application workloads that use Astra Data Store preview as the storage backend
- Trident bindings that use Astra Data Store preview 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 preview cluster and control plane

Follow the steps below to uninstall Astra Data Store preview manually.

Delete the volumes and export policies

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

> 💡 If you do not first delete volumes and export policies, the cluster deletion process pauses until the Astra Data Store preview 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 preview cluster

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

> ℹ️ 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 preview 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 preview license:

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

Delete the Astra Data Store preview installation

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

1. Delete the Astra Data Store preview 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 preview 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 preview 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
```

> (i) Ignore "resources not found errors" in these steps.

3. Remove `/var/lib/kubelet/config.yaml` from all Kubernetes nodes.

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 preview uninstall process

The Astra Data Store preview uninstall process in Kubernetes v1.20 can occasionally cause pods to remain in a terminating state.

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
```

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 preview 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 preview. Case creation using the NetApp Support Site (NSS) is not available for the preview 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 preview.

- 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 preview 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 preview 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 preview 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 preview 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 preview 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-astrads 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 astrads asup collect -c storage -t 2021-01-01T15:00:00Z -d
  24
  ```

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

```
kubectl astrads 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 astrasds 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-astrasds 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 preview 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 astrasds 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>
    ```

    > (i) 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)

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 21.12 release