Installation overview

Astra Data Store

NetApp
February 16, 2022

Table of Contents

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.

   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.

   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.

        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.

        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. 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.

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.

   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.

        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.

        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/
```

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
```

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.