NetApp

Use Astra Data Store

Astra Data Store

NetApp
June 15, 2022

Table of Contents

Use Astra Data Store

Manage Astra Data Store resources with kubectl commands

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

To learn how to deploy a sample app, see Deploy a test application.

For cluster maintenance information, see Manage the cluster.

What you'll need

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

List Kubernetes custom API resources for Astra Data Store

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

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

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

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

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

Response:

```
NAME                         SHORTNAMES  APIGROUP                   NAMESPACED  KIND
astradsautosupports     adsas   astrads.netapp.io/v1alpha1 true
AstraDSAutoSupport
astradscloudsnapshots   adscs   astrads.netapp.io/v1alpha1 true
AstraDSCloudSnapshot
astradsclusters         adscl   astrads.netapp.io/v1alpha1 true
AstraDSCluster
astradsexportpolicies   adsep   astrads.netapp.io/v1alpha1 true
AstraDSExportPolicy
astradsfaileddrives     adsfd   astrads.netapp.io/v1alpha1 true
AstraDSFailedDrive
astradslicenses         adsli   astrads.netapp.io/v1alpha1 true
AstraDSLicense
astradsnfsoptions       adsnf   astrads.netapp.io/v1alpha1 true
AstraDSNfsOption
astradsnodeinfoes       adsni   astrads.netapp.io/v1alpha1 true
AstraDSNodeInfo
astradsnodemanagements adsnm   astrads.netapp.io/v1alpha1 true
AstraDSNodeManagement
astradsqospolicies      adsqp   astrads.netapp.io/v1alpha1 true
AstraDSQosPolicy
astradsversions         adsve   astrads.netapp.io/v1alpha1 true
AstraDSVersion
astradsvolumefiles      adsvf   astrads.netapp.io/v1alpha1 true
AstraDSVolumeFiles
astradsvolumes          adsvo   astrads.netapp.io/v1alpha1 true
AstraDSVolume
astradsvolumesnapshots adsvs   astrads.netapp.io/v1alpha1 true
AstraDSVolumeSnapshot
```

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

```
kubectl get ads -A
```

Response:

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


NAMESPACE         NAME
STATUS      VERSION        SERIAL NUMBER    MVIP           AGE
```

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

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

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

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

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

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

```
NAMESPACE           NAME                                   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

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

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

```
kubectl astrads -h
```

Response:

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

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

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

Flags:
      --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.

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

What you'll need

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

Steps

1. Add a Helm repo from Bitnami:

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

2. Deploy MongoDB:

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

3. Check the status of the MongoDB pod:

```
~% kubectl get pods -n ns-mongodb
NAME                          READY    STATUS    RESTARTS    AGE
mongodb-9846ff8b7-rfr4r    1/1      Running   0           67s
```

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

```
~% kubectl get pvc -n ns-mongodb
NAME        STATUS    VOLUME                          CAPACITY    ACCESS MODES
STORAGECLASS    AGE
mongodb    Bound     pvc-1133453a-e2f5-48a5   8Gi          RWO
trident-csi      97s
```

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

```
~% kubectl get astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-system
NAME                          SIZE         IP             CLUSTER     CREATED
pvc-1133453a-e2f5-48a5    8830116Ki    10.192.2.192   jai-ads      true
```

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

```
~% kubectl describe astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-
system
Name:           pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Namespace:      astrads-system
Labels:         astrads.netapp.io/cluster=jai-ads
                astrads.netapp.io/mip=10.192.1.39
                astrads.netapp.io/volumeUUID=cf33fd38-a451-596c-b656-
61b8270d2b5e
                trident.netapp.io/cloud=on-prem
                trident.netapp.io/creator=trident-dev
                trident.netapp.io/performance=premium
Annotations:  provisioning: {"provisioning":{"cloud":"on-
prem","creator":"trident-dev","performance":"premium"}}
                trident:
                    {"trident":{"version":"21.10.0-test.jenkins-trident-
stable-v21.10-
2+e03219ce37294d9ba54ec476bbe788c1a7772548","backendUUID":"","platform":
```

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

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

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

Manage the Astra Data Store cluster

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

- Add a node
- Remove a node
- Place a node in maintenance mode
- Add drives to a node
- Replace a drive

What you'll need

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

Add a node

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

ⓘ   To add a node using Astra Control Center, see Add nodes to a storage backend cluster.

Steps

1. If the new node's dataIP is not already part of the Astra Data Store cluster CR, do the following:

   a. Edit the cluster CR and add the additional dataIP in the `adsDataNetworks` **addresses** field. Replace the information in capital letters with the appropriate values for your environment:

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

   Response:

   ```
   adsDataNetworks:
       -addresses:  dataIP1,dataIP2,dataIP3,dataIP4,NEW_DATA_IP
   ```

   b. Save the CR.

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

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

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

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

3. Verify that the node has been added:

   ```
   kubectl astrads nodes list
   ```

Remove a node

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

Steps

1. List all the nodes:

```
kubectl astrads nodes list
```

Response:

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

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

```
kubectl astrads nodes remove NODE_NAME
```

Response:

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

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

3. Verify the `present` status of the removed node:

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

Response:

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

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

```
kubectl astrads nodes uninstall NODE_NAME
```

5. Verify the node is removed from the cluster:

```
kubectl astrads nodes list
```

Result

The node is removed from Astra Data Store.

Place a node in maintenance mode

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

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

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

Steps

1. Display the node details:

   ```
   kubectl get nodes
   ```

   Response:

   ```
   NAME             STATUS   ROLES                 AGE      VERSION
   nhcitjj1525      Ready    <none>                3d18h    v1.20.0
   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. Replace the information in capital letters with the appropriate values for your environment:

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

   For example:

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

Response:

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

4. List the nodes:

```
kubectl astrads nodes list
```

Response:

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

5. Check the status of the maintenance mode:

```
kubectl astrads maintenance list
```

Response:

```
NAME      NODE NAME         IN MAINTENANCE   MAINTENANCE STATE
MAINTENANCE VARIANT
node4    nhcitjj1525       true             ReadyForMaintenance       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
```

Add drives to a node

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

What you'll need

- One or more drives that meet the following criteria:

    ◦ Already installed in the node (physical drives) or added to the node VM (virtual drives)

    ◦ No partitions on the drive

    ◦ Drive is not currently in use by the cluster

    ◦ Drive raw capacity does not exceed licensed raw capacity in the cluster (For example, with a license granting 2TB of storage per CPU core, a cluster of 10 nodes would have a maximum of 20TB raw drive capacity)

    ◦ Drive is at least the size of other active drives in the node

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

Steps

1. Describe the cluster:

   ```
   kubectl astrads clusters list
   ```

   Response:

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

2. Note the cluster name.

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

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

   Response:

```
Node: node1.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c290e16d52479a9af5eac c290e16d52479a9af5eac 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c2935798df68355dee0be c2935798df68355dee0be 0
100 GiB false

Node: node2.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.

Node: node3.name
Add drive maximum size: 100.0 GiB
Add drive minimum size: 100.0 GiB
NAME IDPATH SERIAL PARTITIONCOUNT SIZE ALREADYINCLUSTER
sdg /dev/disk/by-id/scsi-3c29ee82992ed7a36fc942 c29ee82992ed7a36fc942 0
100 GiB false
sdh /dev/disk/by-id/scsi-3c29312aa362469fb3da9c c29312aa362469fb3da9c 0
100 GiB false

Node: node4.name
Add drive maximum size: 66.7 GiB
Add drive minimum size: 100.0 GiB
No suitable drives to add exist.
```

4. Do one of the following:

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

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

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

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

Result

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

Replace a drive

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

Example of cluster showing failed drive in nodeStatuses.driveStatuses

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

Response:

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

Example of new AstraDSFailedDrive CR

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

```
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.lab.org
   state: ReadyToReplace
```

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

Response:

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

Steps

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

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

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

   Response:

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

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

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

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

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

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

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

For more information

- Manage Astra Data Store resources with kubectl commands
- Add nodes to a storage backend cluster in Astra Control Center

Monitor Astra Data Store

Monitor metrics with Cloud Insights

You can monitor Astra Data Store metrics using Cloud Insights.

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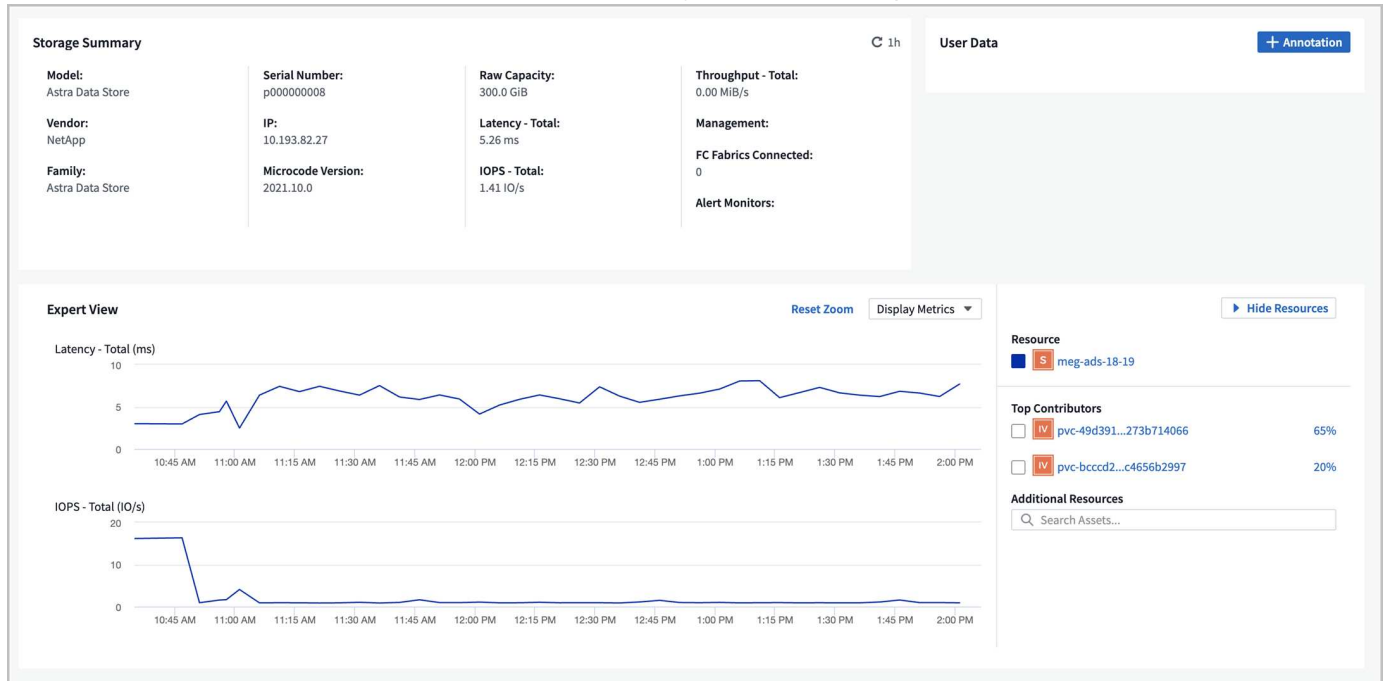
You can also display a list of metrics generated in Astra Data Store by using the Open Metrics API help.

You can complete the following tasks:

- Complete Cloud Insights connection prerequisite tasks
- Acquisition Unit storage
- Download and run the installation script
- Edit the Cloud Insights connection
- Disconnect from Cloud Insights

Complete Cloud Insights connection prerequisite tasks

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

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

Gather the following information:

- **Cloud Insights API access token** with Read/Write permissions to the categories: Acquisition Unit, Data Collection, Data Ingestion and Log Ingestion. This will be used for the read/write operations, setting up the Acquisition Unit, and setting up data ingest processes.
- **Kubernetes API server IP address and port**. This is used to monitor the Astra Data Store cluster.

- **Kubernetes API token**. This is used to call Kubernetes APIs.
- **Persistent volume configuration**. Information about how persistent volumes are provisioned.

Acquisition Unit storage

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

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

Download and run the installation script

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

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

Then, metrics will be sent as follows:

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

Display installer script help

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

Display installer script help text:

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

Response:

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

Run the install script

1. Create a Cloud Insights account, if you don't already have one.
2. Log in to Cloud Insights.
3. From the Cloud Insights menu, click on **Admin** > **Data Collectors**.
4. Click on **+ Data Collector** to add a new collector.
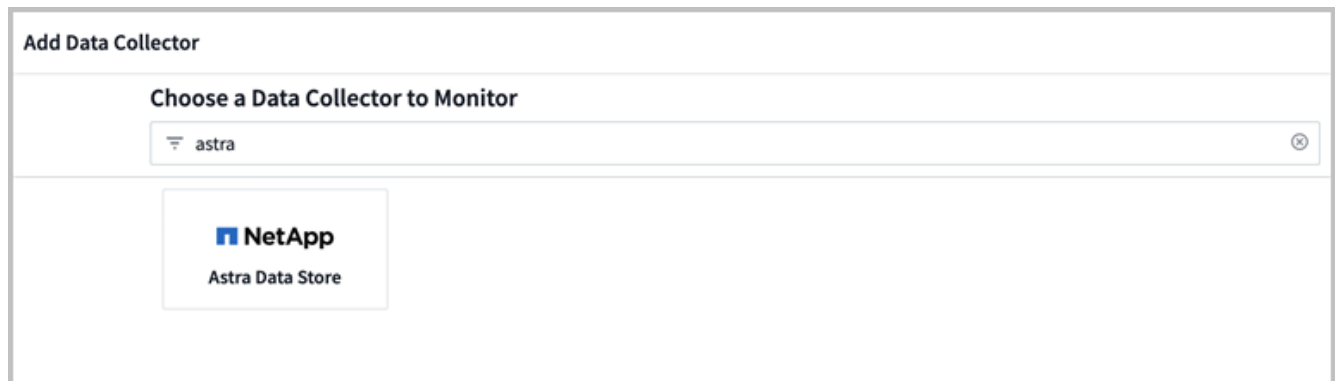
5. Click on the **Astra Data Store** tile.
6. Select the correct Cloud Insights API access token or create a new one.
7. Follow the instructions to download the installer script, update the permissions, and run the script.

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

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 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
 status:
   au-pod-status: UP
   au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9
```

Edit the Cloud Insights connection

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

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

Update the Kubernetes API token

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

Update the Cloud Insights API access token

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

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

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

   The section looks something like this:

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

6. Save and quit the editor window.

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

Disconnect from Cloud Insights

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

Remove the Astra Data Store collector

1. Log in to Cloud Insights.

2. Select **Admin** > **Data Collectors** to access the Data Collectors page.

3. Find the entry for the Astra Data Store cluster.

4. Select the menu on the right side of the screen, and select **Delete**.

5. Click **Delete** on the confirmation page.

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

1. Edit the Agent CR:

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

2. Locate the `au` section and set `isEnabled` to `false`

3. Locate the `fluent-bit` section and remove the plugin named `ads-tail-ci`. If there are no more plugins, you can remove the `fluent-bit` section.

4. If Telegraf is configured, locate the `telegraf` section and remove the plugin named `ads-open-metric`. If there are no more plugins, you can remove the `telegraf` section.

5. Locate the `output-sink` section and remove the sink named `CI`.

6. Save and quit the editor window.

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

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

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

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

8. After the Acquisition Unit pod has been deleted, you can delete the Acquisition Unit from Cloud Insights.

   a. In the Cloud Insights menu, select **Admin** > **Data Collectors**.

   b. Click on the **Acquisition Units** tab.

   c. Click on the menu next to the Acquisition Unit pod.

   d. Select **Delete**.

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

Open Metrics API help

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

- The "HELP" line describes the metric.

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

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

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

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

```
seconds for a volume. Divide by astrads_volume_read_ops to get the average
latency per read operation
# TYPE astrads_volume_read_latency counter
# HELP astrads_volume_read_ops Total number of read operations for a
volume
# TYPE astrads_volume_read_ops counter
# HELP astrads_volume_read_throughput Total read throughput for a volume
in bytes
# TYPE astrads_volume_read_throughput counter
# HELP astrads_volume_total_latency Total accumulated latency in seconds
for all operations on a volume. Divide by astrads_volume_total_ops to get
the average latency per operation
# TYPE astrads_volume_total_latency counter
# HELP astrads_volume_total_ops Total number of operations for a volume
# TYPE astrads_volume_total_ops counter
# HELP astrads_volume_total_throughput Total 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 metrics with Prometheus and Grafana. You can configure Prometheus to
gather metrics from the Astra Data Store Kubernetes cluster metrics endpoint, and you can use Grafana to
visualize the metrics data.

What you'll need

- Make sure that you have downloaded and installed the Prometheus and Grafana packages on the Astra
  Data Store cluster or a different cluster that can communicate with the Astra Data Store cluster. Follow the
  instructions in the official documentation to install each tool:

  - Install Prometheus

  - Install Grafana

- Prometheus and Grafana need to be able to communicate with the Astra Data Store Kubernetes cluster. If
  Prometheus and Grafana are not installed on the Astra Data Store cluster, you need to make sure they can
  communicate with the metrics service running on the Astra Data Store cluster.

Configure Prometheus

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

Steps

1. Edit the `prometheus.yml` configuration file for your Prometheus installation.

2. Add a service target that points to the Astra Data Store service name and its port. For example:

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

3. Start the Prometheus service.

Configure Grafana

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

Steps

1. Edit the `datasources.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 includes Grafana dashboard template files that you can import from within Grafana. These dashboard templates can help you see the types of metrics that are available from Astra Data Store and how you can view them.

Steps

1. Open the Astra Data Store `.tar.gz` bundle.

2. Open the `manifests` directory.

3. Extract the `grafana_cluster.json` and `grafana_volume.json` files.

4. Using the Grafana web UI, import the dashboard template files in to Grafana.

Configure and monitor event logs

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

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

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

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

Enter:

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

where:

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

Set up Cloud Insights

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

Stream event logs to Elastic

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

Enter:

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

ⓘ   The Elastic host name can be an IP address.

Secure Astra Data Store

Manage security certificates

Astra Data Store uses Mutual Transport Layer Security (mTLS) encryption between the cluster's software components. Each Astra Data Store cluster has a self-signed root CA certificate (`astrads-cert-root`) and an intermediate CA certificate (`astrads-cert-<cluster_name>`).
These certificates are managed by the Astra Data Store operator; the operator automatically renews each certificate 7 days before their expiration date. You can also revoke the certificates manually.

Revoke a certificate

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

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

Steps

1. Log in to the controller node in the Astra Data Store cluster.

2. List the existing certificates on the system. For example:

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

   The output should be similar to the following:

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

3. In the output, note the name of the certificate you need to revoke.

4. Use the `kubectl` utility to revoke the certificate, replacing `CERTIFICATE_NAME` with the name of the certificate. For example:

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

Result

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

Manage external keys

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

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

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

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

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

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

Set up external key management

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

What you'll need

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

Steps

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

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

2. Log in to one of the nodes in the Astra Data Store cluster.

3. Configure the key provider for the Astra Data Store cluster by entering the following kubectl extension command:

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

Example

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

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

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

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

Response:

```
Configure SEARS in AstraDS cluster

Usage:
  astrads clusters sears [flags]

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


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

Example

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

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

Check the software encryption at rest status

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

Step

1. Inspect the AstraDSCluster CR.

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

Change external to internal key management

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

Steps

1. Change the AstraDSCluster CR by removing the SoftwareEncryptionAtRest configuration.

2. (Optional) Delete the previous AstraDSKeyProvider and associated secret.

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

Rotate keys for security

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

Configure automatic key rotation

1. Update the AstraDSSEARKeyRotate parameter in the CRD.

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

Configure on-demand key rotation

1. Create an AstraDSSEARKeyRotateRequest CR to rotate keys.

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

Update the Astra Data Store license

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

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

Update the Astra Data Store license using the command line

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

Steps

1. Apply the replacement NetApp License File (NLF) that you obtained from NetApp. Before you run the command, enter the name of the cluster (`<Astra-Data-Store-cluster-name>`) and the path to the license file (`<file_path/file.txt>`):

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

2. Verify that the license has been added:

```
kubectl astrads license list
```

You should see a response similar to the following:

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

Upgrade Astra Data Store

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

Upgrade Astra Data Store using kubectl

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

Download the Astra Data Store bundle and extract the images

Steps

1. Log in to the [NetApp Support Site](#) and download the Astra Data Store bundle (`Astra_Data_Store_2022.05.tar`).

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

   ```
   openssl dgst -sha256 -verify Astra_Data_Store_2022.05.pub -signature
   Astra_Data_Store_2022.05.sig 2022.12.01_ads.tar
   ```

3. Create a directory:

   ```
   mkdir Astra_Data_Store_2022.05
   cd Astra_Data_Store_2022.05
   ```

4. Extract the images:

   ```
   tar -vxzf <path to tar file>/Astra_Data_Store_2022.05.tar
   ```

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

Copy the binary and push images to your local registry

Steps

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

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

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

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

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

   a. Log in to your registry:

   ```
   docker login [your_registry_path]
   ```

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

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

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

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

Perform the upgrade

Steps

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

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

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

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

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

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

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

Uninstall Astra Data Store with an automated script

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

Uninstalling can be done using different methods:

- Uninstall Astra Data Store with an automated script
- Uninstall Astra Data Store manually without a script
- Troubleshoot the Astra Data Store uninstall process

Uninstall Astra Data Store with an automated script

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

What you'll need

- Root administrative permissions

About this task

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

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

Remove existing workloads and bindings

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

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

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

Uninstall Astra Data Store cluster

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

1. Locate the `uninstall.sh` in the `manifests` directory.

2. Run the following `sed` command:

   ```
   sed -i -e 's~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 on Red Hat OpenShift Container Platform (OCP), you can uninstall OCP security context constraints (SCC) and rolebindings resources.

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

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

1. Remove SCC resources:

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

2. Remove rolebindings resources:

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

ⓘ | Ignore "resources not found" errors in these steps.

Uninstall Astra Data Store manually without a script

This process uninstalls Astra Data Store manually without a script.

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

What you'll need

- Root administrative permissions

About this task

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

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

Remove existing workloads and bindings

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

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

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

Uninstall the Astra Data Store cluster and control plane

Follow the steps below to uninstall Astra Data Store manually.

Delete the volumes and export policies

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

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

Steps

1. Delete the volumes:

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

2. Delete the export policies:

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

Delete the Astra Data Store cluster

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

> 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 cluster-scoped resources starts.

1. Delete the cluster:

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

Delete the license

1. ssh to each worker node in the cluster and validate that `firetap` or `netwd` are not running in the worker nodes.

2. Delete the Astra Data Store license:

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

Delete the Astra Data Store installation

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

1. Delete the Astra Data Store installation object:

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

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

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

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

3. Delete remaining artifacts and the operator yaml file:

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

Validate the removal of the astrads-system namespace

Ensure that the following command returns no result:

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

Ensure containers are not running on worker nodes

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

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

Delete OpenShift Container Platform resources

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

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

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

1. Remove SCC resources:

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

2. Remove rolebindings resources:

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

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

Manual deletion sample

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

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

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

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

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

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

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

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

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

$ kubectl get ns | grep astrads-system

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

Troubleshoot the Astra Data Store uninstall process

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

Pods in terminating state

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

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

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

Quality of Service policy points to old cluster

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

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

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

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

   ```
   uninstall.sh all
   ```

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

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

Details

Use the following workaround instructions:

Steps

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

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

   Response:

   ```
   NAME                    AGE
   externalkeyprovider1    94s
   ```

2. Remove the key provider CRs:

   a. Remove the finalizer:

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

   b. Remove the finalizer line highlighted below:

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

```
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSKeyProvider
metadata:
  creationTimestamp: "2022-05-24T16:38:27Z"
  finalizers:
  - astrads.netapp.io/astradskeyprovider-finalizer
  generation: 1
  labels:
    astrads.netapp.io/cluster: astrads-cluster-example
    astrads.netapp.io/rsid: "1"
  name: externalkeyprovider1
  namespace: astrads-system
  resourceVersion: "1134699"
  uid: a11111b2-31c0-4575-b7f3-97f9abla1bla
spec:
  cluster: astrads-cluster-example
  kmipServer:
    hostnames:
    - 10.xxx.xxx.xxx
    port: 5696
    secretRef: externalkeyprovider1
status:
  keyProviderUUID: a1b2cd34-4fc6-5bae-9184-2288c673181d
  kmipServerStatus:
    capabilities: '{ KMIP_library_version()=17367809,
KMIP_library_version_str()="KMIP
      1.9.3a  8-Apr-2019", KMIP_library_version_tag()="KMIP part
of KMIP 1.9.3a  8-Apr-2019",
      KMIP_library_is_eval()=false,
KMIP_library_fips_capable()=true(FIPS140),
KMIP_SSL_provider_build_version()=268444095,
      KMIP_SSL_provider_version()=268444095,
KMIP_SSL_provider_version_str()="OpenSSL
      1.0.2zb-fips  23 Sep 2021" }'
    keyServerUUID: 8422bdd0-74ad-579d-81bd-6d544ac4224a
```

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

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