



# **Astra Data Store**를 사용하십시오

## Astra Data Store

NetApp  
June 06, 2022

# 목차

Astra Data Store를 사용하십시오 . . . . .	1
kubeck 명령을 사용하여 Astra Data Store 리소스를 관리합니다 . . . . .	1
테스트 응용 프로그램을 배포합니다 . . . . .	7
Astra Data Store 클러스터를 관리합니다 . . . . .	11
Astra Data Store를 모니터링합니다 . . . . .	21
보안 Astra 데이터 저장소 . . . . .	36
Astra Data Store 라이선스를 업데이트합니다 . . . . .	40
Astra Data Store를 업그레이드합니다 . . . . .	41
자동 스크립트로 Astra Data Store를 제거합니다 . . . . .	43

# Astra Data Store를 사용하십시오

## kubeck 명령을 사용하여 Astra Data Store 리소스를 관리합니다

kubtl 명령을 사용하거나 Kubernetes API 확장을 사용하여 Astra Data Store 리소스를 관리할 수 있습니다.

샘플 앱을 배포하는 방법에 대한 자세한 내용은 을 참조하십시오 ["테스트 응용 프로그램을 배포합니다"](#).

클러스터 유지 보수 정보는 를 참조하십시오 ["클러스터를 관리합니다"](#).

무엇을 '필요로 할거야

- 에 설치한 Astra Data Store kubeck 플러그인 ["Astra Data Store를 설치합니다"](#)

## Astra Data Store용 Kubernetes 사용자 지정 API 리소스를 나열합니다

Kubernetes 내의 kubtl 명령을 사용하여 Astra Data Store 클러스터와 상호 작용하고 상태를 관찰할 수 있습니다.

'api-resources' 명령에 나열된 각 항목은 Astra Data Store에서 클러스터 관리를 위해 내부적으로 사용하는 Kubernetes 사용자 정의 리소스 정의(CRD)를 나타냅니다.

이 목록은 나중에 보여진 것처럼 입력을 줄이기 위해 각 Astra Data Store 개체의 짧은 이름을 얻는 데 특히 유용합니다.

1. Astra Data Store용 Kubernetes 사용자 API 리소스 목록을 표시합니다.

```
kubect1 api-resources --api-group astrads.netapp.io
```

응답:

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

2. Kubernetes 클러스터의 모든 현재 Astra Data Store 객체를 가져오려면 "kubect get ads-a" 명령을 사용하십시오.

```
kubect1 get ads -A
```

응답:

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. 간단한 이름 중 하나를 사용하여 클러스터에 있는 볼륨의 현재 상태를 표시합니다.

```
kubectl get adsvo -A
```

응답:

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				

## kubeck 확장명에 대한 도움말 옵션을 사용합니다

NetApp은 라이선스 추가, 노드 관리, 문제 해결, Astra Data Store 클러스터 확장 등의 Astra Data Store 클러스터 관리 작업을 수행할 수 있도록 kubectl 명령에 대한 "astrs"를 확장하였습니다. 이 확장에는 내선사용 방법과 작업에 대한 정보를 제공하는 '-h' 옵션이 포함되어 있습니다.

1. Astra Data Store의 kubectl 확장명에 대한 모든 명령어 도움말을 보여줌:

```
kubectl astrads -h
```

응답:

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

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

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

#### Flags:

<code>--as</code> string	Username to impersonate for the operation
<code>--as-group</code> stringArray	Group to impersonate for the operation, this flag can be repeated to specify multiple groups.
<code>--cache-dir</code> string	Default HTTP cache directory (default "/u/arda/.kube/http-cache")
<code>--certificate-authority</code> string	Path to a cert file for the certificate authority
<code>--client-certificate</code> string	Path to a client certificate file for TLS
<code>--client-key</code> string	Path to a client key file for TLS
<code>--cluster</code> string	The name of the kubeconfig cluster to use
<code>--context</code> string	The name of the kubeconfig context to use
<code>-h, --help</code>	help for astrads
<code>--insecure-skip-tls-verify</code>	If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure
<code>--kubeconfig</code> string	Path to the kubeconfig file to use for CLI requests.
<code>-n, --namespace</code> string	If present, the namespace scope for this CLI request
<code>--request-timeout</code> 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).
<code>-s, --server</code> string	The address and port of the Kubernetes API server
<code>--token</code> string	Bearer token for authentication to the API server
<code>--user</code> string	The name of the kubeconfig user

to use

2. 명령에 대한 자세한 내용은 'astrads[command]--help'를 사용하십시오.

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

응답:

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

Usage:

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

Examples:

```
# Control plane collection
```

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

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

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

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

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

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

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

NOTE:

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

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

Flags:

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

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

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

```
-f, --forceUpload           Configure an AutoSupport to upload if  
it is in the compressed state  
and not
```



the 'local' option or if	uploading because it was created with
disabled	automatic uploads of AutoSupports is
	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
	Specify nodes to collect for storage
--nodes string	
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")	

## 테스트 응용 프로그램을 배포합니다

다음은 Astra Data Store에서 사용할 수 있는 테스트 애플리케이션을 구축하는 단계입니다.

이 예에서는 Hrom 리포지토리를 사용하여 Bitnami의 MongoDB 차트를 배포합니다.

무엇을 '필요로 할거야

- Astra Data Store 클러스터가 구축 및 구성되었습니다
- Trident 설치가 완료되었습니다

단계

1. Bitnami에서 Helm repo 추가:

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

2. MongoDB 구축:

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

### 3. MongoDB Pod의 상태를 확인합니다.

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

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

### 4. MongoDB에서 사용되는 영구 볼륨 클레임(PVC)을 확인합니다.

```
~% 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. kubelet 명령 'Get astradsvolume'을 사용하여 볼륨을 나열합니다.

```
~% 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. kubect describe astradsvolume을 사용하여 볼륨을 설명합니다.

```
~% 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

```

## Astra Data Store 클러스터를 관리합니다

Astra Data Store에서 `kubtl` 명령을 사용하여 클러스터를 관리할 수 있습니다.

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

무엇을 '필요로 할거야

- kubectl 및 kubectl-astrads 플러그인이 설치된 시스템. 을 참조하십시오 ["Astra Data Store를 설치합니다"](#).

## 노드를 추가합니다

추가하려는 노드는 Kubernetes 클러스터의 일부여야 하며 클러스터의 다른 노드와 비슷한 구성을 가져야 합니다.



Astra Control Center를 사용하여 노드를 추가하려면 을 참조하십시오 ["스토리지 백엔드 클러스터에 노드를 추가합니다"](#).

### 단계

1. 새 노드의 데이터 IP가 Astra Data Store 클러스터 CR에 아직 포함되어 있지 않은 경우 다음을 수행합니다.
  - a. 클러스터 CR을 편집하고 "adsDataNetworks" \* Addresses \* 필드에 추가 데이터 IP를 추가합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

응답:

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

- a. CR을 저장합니다.
- b. Astra Data Store 클러스터에 노드를 추가합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

2. 그렇지 않으면 노드를 추가하면 됩니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

3. 노드가 추가되었는지 확인합니다.

```
kubectl astrads nodes list
```

## 노드를 제거합니다

Astra Data Store와 함께 kubectl 명령을 사용하여 클러스터의 노드를 제거합니다.

### 단계

1. 모든 노드 나열:

```
kubectl astrads nodes list
```

응답:

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

2. 제거할 노드를 표시합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes remove NODE_NAME
```

응답:

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

노드를 제거하도록 표시한 후 노드 상태가 활성에서 현재로 바뀌어야 합니다.

3. 제거된 노드의 'Present' 상태를 확인한다.

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

응답:

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

4. 노드에서 Astra Data Store를 제거합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

```
kubectl astrads nodes uninstall NODE_NAME
```

5. 클러스터에서 노드가 제거되었는지 확인합니다.

```
kubectl astrads nodes list
```

노드가 Astra Data Store에서 제거됩니다.



## 노드를 유지보수 모드로 전환합니다

호스트 유지보수 또는 패키지 업그레이드를 수행해야 하는 경우 노드를 유지보수 모드로 전환해야 합니다.



노드는 이미 Astra Data Store 클러스터에 포함되어 있어야 합니다.

노드가 유지보수 모드일 때는 클러스터에 노드를 추가할 수 없습니다. 이 예제에서는 노드 "nhcitj1525"를 유지보수 모드로 설정합니다.

단계

1. 노드 세부 정보를 표시합니다.

```
kubectl get nodes
```

응답:

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. 노드가 유지보수 모드가 아닌지 확인합니다.

```
kubectl astrads maintenance list
```

응답(유지보수 모드에 있는 노드가 이미 없음):

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

3. 유지보수 모드를 활성화합니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

예를 들면 다음과 같습니다.

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

응답:

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

#### 4. 노드 나열:

```
kubectl astrads nodes list
```

응답:

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

#### 5. 유지보수 모드의 상태를 점검합니다.

```
kubectl astrads maintenance list
```

응답:

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

유지보수 모드는 거짓으로 시작해 참으로 바뀝니다. 유지 보수 상태가 PreparingForMaintenance에서 ReadyforMaintenance로 바뀝니다.

#### 6. 노드 유지보수가 완료된 후 유지보수 모드를 비활성화합니다.

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

#### 7. 노드가 더 이상 유지보수 모드가 아닌지 확인합니다.

```
kubectl astrads maintenance list
```

## 노드에 드라이브를 추가합니다

Astra Data Store와 함께 kubectl 명령을 사용하여 Astra Data Store 클러스터의 노드에 물리적 또는 가상 드라이브를 추가합니다.

무엇을 '필요로 할거야

- 다음 기준을 충족하는 하나 이상의 드라이브:
  - 노드에 이미 설치되었거나(물리적 드라이브) 노드 VM(가상 드라이브)에 추가되었습니다.
  - 드라이브에 파티션이 없습니다
  - 드라이브가 현재 클러스터에서 사용되고 있지 않습니다
  - 드라이브 물리적 용량이 클러스터의 라이선스 물리적 용량을 초과하지 않음(예: CPU 코어당 2TB의 스토리지를 라이선스를 부여하면 10개 노드로 구성된 클러스터에 최대 20TB의 물리적 드라이브 용량이 있음)
  - 드라이브가 노드에 있는 다른 활성 드라이브의 크기입니다



Astra Data Store에는 노드당 드라이브를 16개 이상 필요로 하지 않습니다. 17번째 드라이브를 추가하려고 하면 드라이브 추가 요청이 거부됩니다.

단계

### 1. 클러스터 설명:

```
kubectl astrads clusters list
```

응답:

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

### 2. 클러스터 이름을 기록합니다.

### 3. 클러스터의 모든 노드에 추가할 수 있는 드라이브를 표시합니다. cluster\_name을 클러스터 이름으로 바꿉니다.

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

응답:

```

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. 다음 중 하나를 수행합니다.

- 사용 가능한 모든 드라이브의 이름이 동일한 경우 해당 노드에 동시에 추가할 수 있습니다. 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

- 드라이브의 이름이 다른 경우 각 노드에 하나씩 추가할 수 있습니다(추가해야 하는 각 드라이브에 대해 이 단계를 반복해야 함). 대문자에서 정보를 환경에 적합한 값으로 바꿉니다.

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

Astra Data Store가 드라이브 추가 요청을 생성하고 요청 결과와 함께 메시지가 나타납니다.

## 드라이브를 교체합니다

클러스터에서 드라이브가 고장난 경우 데이터 무결성을 보장하기 위해 가능한 한 빨리 드라이브를 교체해야 합니다. 드라이브에 장애가 발생하면 클러스터 CR 노드 상태, 클러스터 상태 정보 및 메트릭 끝점에서 장애가 발생한 드라이브에 대한 정보를 볼 수 있습니다. 다음 예제 명령을 사용하여 오류가 발생한 드라이브 정보를 볼 수 있습니다.

노드 **Statuses.driveStatuses**에서 장애가 발생한 드라이브를 표시하는 클러스터의 예

```
kubectl get adsc1 -A -o yaml
```

응답:

```
...
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
  ...
```

장애가 발생한 드라이브 CR은 장애가 발생한 드라이브의 UUID에 해당하는 이름으로 클러스터에 자동으로 생성됩니다.

```
kubectl get adsfd -A -o yaml
```

응답:

```

...
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
```

응답:

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

단계

1. 교체 제한 사항에 맞는 드라이브를 필터링하는 "kubbeck astrads faileddrive show-replacement" 명령을 사용하여 가능한 교체 드라이브를 나열합니다(클러스터에서 사용되지 않음, 마운트되지 않음, 파티션 없음, 오류가 발생한 드라이브보다 크거나 같음).

가능한 대체 드라이브를 필터링하지 않고 모든 드라이브를 나열하려면 'show-replacement' 명령에 '--all'을 추가합니다.

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

응답:

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

2. "replace" 명령을 사용하여 드라이브를 전달된 일련 번호로 교체합니다. 명령이 대체를 완료하거나, '--wait' 시간이 경과되면 실패합니다.

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



부적절한 일련 번호를 사용하여 `kubctl astrads faileddrive replace`를 실행하면 다음과 같은 오류가 나타납니다.

```
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. 드라이브 교체를 다시 실행하려면 이전 명령으로 '--force'를 사용하십시오.

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

를 참조하십시오

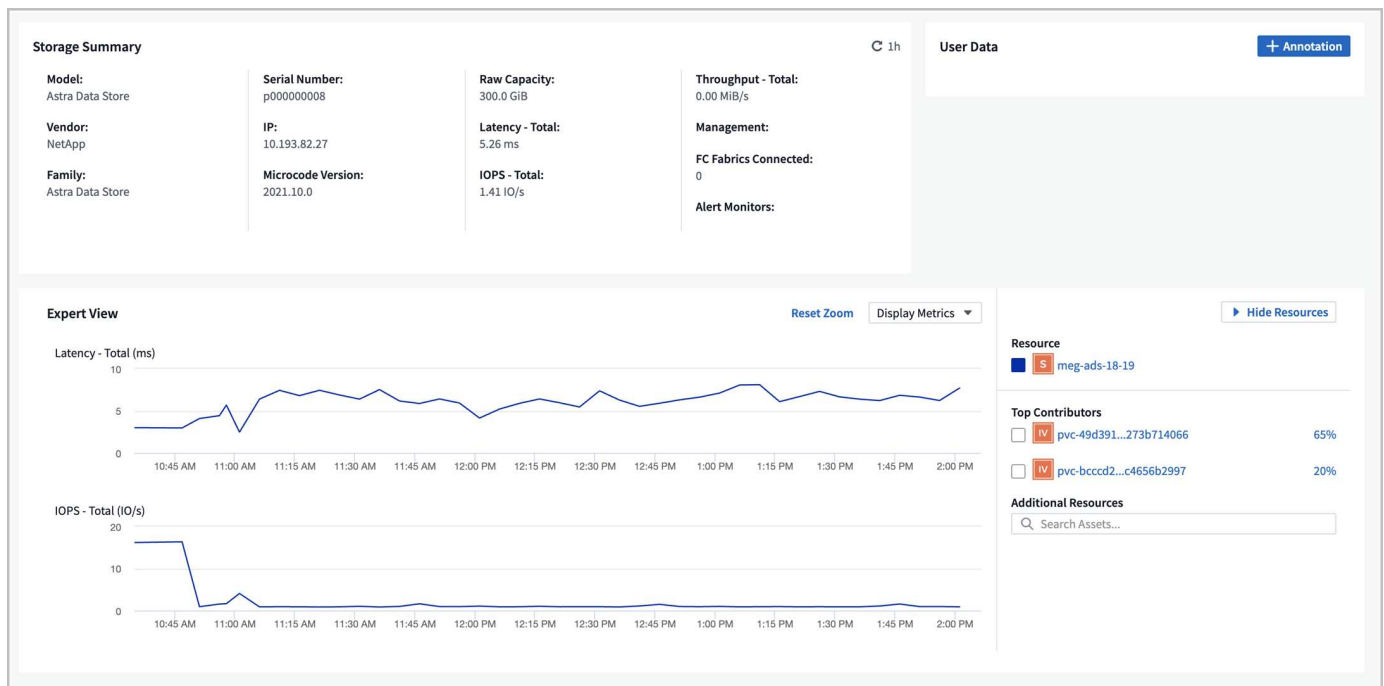
- ["kubect 명령을 사용하여 Astra Data Store 리소스를 관리합니다"](#)

## Astra Data Store를 모니터링합니다

### Cloud Insights를 통해 메트릭을 모니터링합니다

Cloud Insights를 사용하여 Astra 데이터 저장소 메트릭을 모니터링할 수 있습니다.

다음은 Cloud Insights에 표시되는 몇 가지 Astra 데이터 저장소 메트릭입니다



을 사용하여 Astra Data Store에서 생성된 메트릭 목록을 표시할 수도 있습니다 [\[Open Metrics API help\]](#).

다음 작업을 완료할 수 있습니다.

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

## Cloud Insights 연결 필수 작업을 완료합니다

Astra 데이터 저장소를 Cloud Insights와 연결하기 전에 다음 작업을 완료해야 합니다.

- ["Astra Data Store Monitoring Operator를 설치합니다"](#) 이것은 Astra Data Store 설치 지침의 일부입니다.
- ["kubbeck-astrads 바이너리를 설치합니다"](#) 이것은 Astra Data Store 설치 지침의 일부입니다.
- ["Cloud Insights 계정을 만듭니다"](#).
- 'awk, curl, grep', 'jq' 명령을 사용할 수 있는지 확인합니다

다음 정보를 수집합니다.

- 획득 장치, 데이터 수집, 데이터 수집 및 로그 수집 등의 범주에 대한 읽기/쓰기 권한이 있는 \* Cloud Insights API 액세스 토큰 \*. 이 작업은 읽기/쓰기 작업, 획득 장치 설정 및 데이터 수집 프로세스 설정에 사용됩니다.
- \* Kubernetes API 서버 IP 주소 및 포트 \*. Astra Data Store 클러스터를 모니터링하는 데 사용됩니다.
- \* Kubernetes API 토큰 \*. Kubernetes API를 호출하는 데 사용됩니다.



- \* 영구 볼륨 구성 \*. 영구 볼륨의 프로비저닝 방법에 대한 정보입니다.

## 획득 장치 저장

획득 장치는 설치 파일, 구성 데이터 및 로그를 저장하기 위해 세 개의 영구 볼륨이 필요합니다. Monitoring Operator는 기본 스토리지 클래스를 사용하여 영구 볼륨 클레임을 생성합니다. 설치 프로그램 스크립트를 실행할 때 '-s' 옵션을 사용하여 다른 스토리지 클래스 이름을 지정할 수 있습니다.

Kubernetes 클러스터에 스토리지 공급자(예: NetApp Trident)가 없으면 설치 관리자 스크립트를 실행할 때 '-r' 옵션을 사용하여 로컬 파일 시스템 경로를 제공할 수 있습니다. '-r' 옵션이 설정되면 설치 프로그램 스크립트는 제공된 디렉토리 내에 세 개의 영구 볼륨을 생성합니다. 이 디렉토리에는 최소 150GB의 여유 공간이 필요합니다.

## 설치 스크립트를 다운로드하고 실행합니다

Cloud Insights는 모니터링 오퍼레이터를 통해 Astra 데이터 저장소 모니터링을 활성화하는 Bash 스크립트를 제공합니다. 설치 스크립트는 Astra Data Store Collector와 Fluent Bit Agent를 사용하여 획득 장치를 설치합니다.

Cloud Insights 테넌트 도메인 이름과 선택한 Cloud Insights API 액세스 토큰은 다운로드될 때 설치 프로그램 스크립트에 포함됩니다.

그런 다음 다음과 같이 메트릭이 전송됩니다.

- Cloud Insights 획득 장치는 Cloud Insights 데이터 레이크에 메트릭을 전송합니다.
- Fluent bit는 로그 수집 서비스에 로그를 보냅니다.

설치 프로그램 스크립트 도움말을 표시합니다

설치 프로그램 스크립트에 대한 전체 도움말 텍스트가 아래에 나와 있습니다.

설치 프로그램 스크립트 도움말 텍스트 표시:

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

응답:

```

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.

```


설치 스크립트를 실행합니다

1. Cloud Insights 계정이 없는 경우 계정을 만듭니다.
2. Cloud Insights에 로그인합니다.
3. Cloud Insights 메뉴에서 \* Admin \* > \* Data Collector \* 를 클릭합니다.
4. 새 수집기를 추가하려면 \* + Data Collector \* 를 클릭합니다.




5. Astra Data Store \* 타일을 클릭합니다.
6. 올바른 Cloud Insights API 액세스 토큰을 선택하거나 새 토큰을 생성하십시오.
7. 지침에 따라 설치 프로그램 스크립트를 다운로드하고, 권한을 업데이트하고, 스크립트를 실행합니다.


이 스크립트에는 Cloud Insights 테넌트 URL 및 선택한 Cloud Insights API 액세스 토큰이 포함되어 있습니다.



Select a Data Collector



Configure Collector



**NetApp**  
Astra Data Store

## Configure Collector

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

**What Operating System or Platform Are You Using?**

Kubernetes

**Select existing API Access Token or create a new one**

default\_ads\_api\_key1 (...d0gHof)

[+ API Access Token](#)

[Production Best Practices ?](#)

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

- 1

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

Copy Installer Script

☐ Reveal Installer Script

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

CI_DOMAIN_NAME="f49uaky.gstabler-ads.cloudinsights-dev.netapp.com"
CI_API_KEY="eyJraWQ1OiI5OTk5IiwidHlwIjoIc3R5bWVudF9hZHNfYXBPX2t1eTEgKG9uIGJlaGFsZiBvZiBhZG1pbik"

```
- 3

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

Copy Permissions Command

☐ Reveal Permissions Command

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

```
- 5

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

Copy Install Command

☐ Reveal Install Command

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

```
- 7

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

Complete Setup

8. 스크립트가 완료되면 \* 설치 완료 \* 를 클릭합니다.

설치 스크립트가 완료되면 데이터 소스 목록에 Astra Data Store Collector가 나타납니다.



오류로 인해 스크립트가 종료되면 나중에 오류가 해결되면 스크립트를 다시 실행할 수 있습니다. 이 스크립트는 환경에서 기본 설정을 사용하지 않는 경우 모니터링 운영자 네임스페이스 및 Kubernetes API 서버 포트와 같은 추가 매개 변수를 지원합니다. 사용법과 도움말 텍스트를 보려면 의 ``h' 옵션을 사용하십시오./cloudinsights-ads-monitoring.sh -h.

설치 스크립트는 구성이 성공할 때 다음과 같은 출력을 생성합니다.

```
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!
```

상담원 **CR**의 예

다음은 설치 프로그램 스크립트를 실행한 후 Monitoring-NetApp 에이전트 CR이 어떻게 보일지에 대한 예입니다.

```

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

```

## Cloud Insights 연결을 편집합니다

나중에 Kubernetes API 토큰 또는 Cloud Insights API 액세스 토큰을 편집할 수 있습니다.

- Kubernetes API 토큰을 업데이트하려면 Cloud Insights UI에서 Astra Data Store Collector를 편집해야 합니다.
- 원격 측정 및 로그에 사용되는 Cloud Insights API 액세스 토큰을 업데이트하려면 `kubctl` 명령을 사용하여 모니터링 오퍼레이터 CR을 편집해야 합니다.

#### Kubernetes API 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. Admin \* > \* Data Collector \* 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 페이지 오른쪽에 있는 메뉴를 클릭하고 \* 편집 \* 을 선택합니다.
5. Kubernetes API 토큰 필드를 새 값으로 업데이트합니다.
6. Collector 저장 \* 을 선택합니다.

#### Cloud Insights API 액세스 토큰을 업데이트합니다

1. Cloud Insights에 로그인합니다.
2. 관리자 \* > \* API 액세스 \* 를 선택하고 \* + API 액세스 토큰 \* 을 클릭하여 새 Cloud Insights API 액세스 토큰을 만듭니다.
3. 상담원 CR 편집:

```
kubect1 --namespace netapp-monitoring edit agent agent-monitoring-netapp
```

4. 출력 싱크 섹션을 찾아 이름이 'CI'인 항목을 찾습니다.
5. 'api-key'라는 레이블의 경우 현재 값을 새 Cloud Insights API 액세스 토큰으로 바꿉니다.

섹션은 다음과 같이 표시됩니다.

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

6. 편집기 창을 저장하고 종료합니다.

모니터링 운영자는 새로운 Cloud Insights API 액세스 토큰을 사용하도록 Fluent 비트를 업데이트합니다.

#### Cloud Insights와의 연결을 해제합니다

Cloud Insights와의 연결을 끊으려면 먼저 Cloud Insights UI에서 Astra 데이터 저장소 수집기를 삭제해야 합니다. 이 작업이 완료되면 모니터링 작동자가 획득 장치, Telegraf(구성된 경우) 및 Fluent 비트 구성을 제거할 수 있습니다.

#### Astra Data Store Collector를 제거합니다

1. Cloud Insights에 로그인합니다.

2. Admin \* > \* Data Collector \* 를 선택하여 Data Collector 페이지에 액세스합니다.
3. Astra Data Store 클러스터의 항목을 찾습니다.
4. 화면 오른쪽의 메뉴를 선택하고 \* Delete \* 를 선택합니다.
5. 확인 페이지에서 \* 삭제 \* 를 클릭합니다.

획득 장치, **Telegraf**(전신)(구성된 경우) 및 **Fluent bit**를 제거합니다

1. 상담원 CR 편집:

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

2. au 섹션을 찾아 IsEnabled를 false로 설정합니다
3. '유창한 비트' 섹션을 찾아 ads-tail-CI'라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 "fluent-bit" 섹션을 제거할 수 있습니다.
4. 텔레그래프가 구성된 경우 텔레그래프 섹션을 찾아 광고 공개 메트릭이라는 플러그인을 제거합니다. 플러그인이 더 이상 없으면 Telegraf 섹션을 제거할 수 있습니다.
5. 출력 싱크 섹션을 찾아 'CI'라는 싱크를 제거합니다.
6. 편집기 창을 저장하고 종료합니다.

모니터링 오퍼레이터는 Telegraf(Telegraf)(구성된 경우) 및 Fluent 비트 구성을 업데이트하고 획득 장치 포드를 삭제합니다.

7. Storage Provisioner 대신 Acquisition Unit PVS에 로컬 디렉토리를 사용한 경우 PVS를 삭제합니다.

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

그런 다음 획득 장치가 실행 중인 노드에서 실제 디렉토리를 삭제합니다.

8. 획득 장치 포드가 삭제된 후 Cloud Insights에서 획득 장치를 삭제할 수 있습니다.
  - a. Cloud Insights 메뉴에서 \* Admin \* > \* Data Collector \* 를 선택합니다.
  - b. Acquisition Units(획득 단위) \* 탭을 클릭합니다.
  - c. 획득 장치 포드 옆에 있는 메뉴를 클릭합니다.
  - d. 삭제 \* 를 선택합니다.

모니터링 오퍼레이터는 Telegraf(전신)(구성된 경우) 및 Fluent bit 구성을 업데이트하고 획득 장치를 제거합니다.

메트릭 **API** 도움말을 엽니다

다음은 Astra Data Store에서 메트릭을 수집하는 데 사용할 수 있는 API 목록입니다.

- "도움말" 줄에 메트릭이 설명되어 있습니다.
- "유형" 선은 메트릭이 게이지 또는 카운터인지 여부를 나타냅니다.

```

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

```



```

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

```

```

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

```

```

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

```

## Prometheus 및 Grafana를 사용하여 메트릭을 모니터링합니다

Prometheus 및 Grafana를 사용하여 Astra Data Store 메트릭을 모니터링할 수 있습니다. Astra Data Store Kubernetes 클러스터 메트릭 엔드포인트에서 메트릭을 수집하도록 Prometheus를 구성할 수 있으며 Grafana를 사용하여 메트릭 데이터를 시각화할 수 있습니다.

무엇을 '필요로 할거야

- Astra Data Store 클러스터 또는 Astra Data Store 클러스터와 통신할 수 있는 다른 클러스터에 Prometheus 및 Grafana 패키지를 다운로드하여 설치했는지 확인하십시오. 공식 설명서의 지침에 따라 각 도구를 설치합니다.
  - ["Prometheus를 설치합니다"](#)
  - ["Grafana를 설치합니다"](#)
- Prometheus 및 Grafana는 Astra Data Store Kubernetes 클러스터와 통신할 수 있어야 합니다. Prometheus 및 Grafana가 Astra Data Store 클러스터에 설치되어 있지 않은 경우, Astra Data Store 클러스터에서 실행 중인 메트릭 서비스와 통신할 수 있어야 합니다.

## Prometheus를 구성합니다

Astra Data Store는 Kubernetes 클러스터의 TCP 포트 9341에 메트릭 서비스를 제공합니다. 이 서비스에서 메트릭을 수집하려면 Prometheus를 구성해야 합니다.

단계

1. Prometheus 설치를 위해 'Prometheus.yml' 설정 파일을 편집합니다.
2. Astra Data Store 서비스 이름과 해당 포트를 가리키는 서비스 목표를 추가합니다. 예를 들면 다음과 같습니다.

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

3. Prometheus 서비스를 시작합니다.

## Grafana 구성

Grafana를 구성하여 Prometheus에서 수집한 메트릭을 표시할 수 있습니다.

단계

1. Grafana 설치를 위해 'datasources.yaml' 구성 파일을 편집합니다.
2. Prometheus를 데이터 소스로 추가합니다. 예를 들면 다음과 같습니다.

```
apiVersion: 1

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

3. Grafana 서비스를 시작합니다.
4. 에 대한 Grafana 설명서의 지침을 따릅니다 ["시작하십시오"](#).

## Grafana 대시보드 템플릿을 가져옵니다

Astra Data Store를 설치하기 위해 다운로드한 번들 파일에는 Grafana 내에서 가져올 수 있는 Grafana 대시보드 템플릿 파일이 포함되어 있습니다. 이러한 대시보드 템플릿을 사용하면 Astra Data Store에서 사용 가능한 메트릭의 유형과 이를 보는 방법을 확인할 수 있습니다.

단계

1. Astra Data Store의 .tar.gz bundle을 엽니다.
2. 'manifests' 디렉터리를 엽니다.

3. `graana_cluster.json`과 `graana_volume.json` 파일을 추출한다.
4. Grafana 웹 UI 사용, "[에서 대시보드 템플릿 파일을 Grafana로 가져옵니다](#)".

## 이벤트 로그를 구성하고 모니터링합니다

EMS(이벤트 관리 시스템) 로그를 모니터링하려면 다음과 같은 높은 수준의 작업을 수행해야 합니다.

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

### Astra Data Store 클러스터 사용자 지정 리소스(CR)에서 모니터링 구성

Astra Data Store 클러스터 CR에 모니터링 옵션이 구성되어 있지 않은 경우 "Astrads" 확장을 사용하여 설정할 수 있습니다.

입력:

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

여기서,

- 설치된 에이전트의 네임스페이스: 모니터링 에이전트의 기본 이름인 모니터링 에이전트의 네임스페이스를 입력합니다. 모니터링 오퍼레이터의 경우 NetApp CR입니다.
- '-r'은 Fluent 또는 Telegraf 이미지가 있는 Docker 레지스트리를 설정하는 데 선택 사항입니다. 기본적으로 경로는 변경할 수 있는 `dddocker.repo.eng.netapp.com/global/astra`` 로 설정됩니다.

### Cloud Insights를 설정합니다

로그를 보려면 Cloud Insights 설정은 선택 사항이지만 Cloud Insights를 사용하여 데이터를 보는 것이 좋습니다. 을 참조하십시오 "[NetApp Cloud Insights 설정 방법](#)" Astra Data Store와 함께 사용

이벤트 로그를 **Elastic**로 스트리밍합니다

EMS 이벤트 및 기타 POD 로그를 Elastic 등의 타사 엔드포인트로 스트리밍하려면 "Astrads" 확장을 사용합니다.

입력:

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



탄성 호스트 이름은 IP 주소일 수 있습니다.

# 보안 Astra 데이터 저장소

## 보안 인증서를 관리합니다

Astra Data Store는 클러스터의 소프트웨어 구성 요소 간에 MTL(Mutual Transport Layer Security) 암호화를 사용합니다. 각 Astra Data Store 클러스터에는 자체 서명된 루트 CA 인증서("astrads-cert-root")와 중간 CA 인증서("astrads-cert-<cluster\_name>")가 있습니다. 이 인증서는 Astra Data Store 운영자가 관리합니다. 운영자는 만료 날짜 7일 전에 각 인증서를 자동으로 갱신합니다. 인증서를 수동으로 취소할 수도 있습니다.

### 인증서를 해지합니다

Astra Data Store 컨트롤러, 노드 또는 CA 인증서가 손상된 경우 MTL 암호를 삭제하여 이를 취소할 수 있습니다. 이렇게 하면 Astra Data Store 운영자가 자동으로 새 인증서를 발급합니다. 언제든지 Astra Data Store 인증서를 해지할 수 있습니다.



CA 인증서를 해지하면 해당 CA에서 서명한 인증서가 해지됩니다.

### 단계

1. Astra Data Store 클러스터의 컨트롤러 노드에 로그인합니다.
2. 시스템에 있는 기존 인증서를 나열합니다. 예를 들면 다음과 같습니다.

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

출력은 다음과 비슷해야 합니다.

```
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. 출력에서 취소할 인증서의 이름을 기록합니다.
4. kubectl 유틸리티를 사용하여 인증서를 해지하고 certificate\_name을 인증서 이름으로 바꿉니다. 예를 들면 다음과 같습니다.

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

기존 인증서가 해지되고 대신 새 인증서가 자동으로 생성됩니다.

## 외부 키 관리

하나 이상의 외부 키 관리 서버를 사용하여 클러스터가 암호화된 데이터에 액세스하는 데 사용하는 키를 보호할 수 있습니다. 외부 키 관리 서버는 KMIP(Key Management Interoperability Protocol)를 사용하여 노드에 키를 제공하는 스토리지 환경의 타사 시스템입니다.



Astra Data Store는 Astra Data Store 클러스터를 생성할 때 기본적으로 내부 키 제공업체와 함께 Software Encryption at Rest(sear)를 활성화합니다.

키 관리에는 다음과 같은 사용자 정의 리소스 정의(CRD)가 포함됩니다.

- \* AstraDSKeyProvider \*: 외부 KMIP 서버를 구성합니다. 이는 서버 클러스터일 수 있습니다.
- \* AstraDSSEARKeyRotate \*: 키 공급자에서 새 키 암호화 키를 가져옵니다. 이 키를 Astra Data Store에 제공합니다.

외부 키 관리와 관련된 다음 작업을 수행할 수 있습니다.

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

외부 키 관리를 설정합니다

Astra Data Store에서 외부 키 관리를 설정하는 것은 kubectl asts 명령어를 사용한다.

예를 들어, OpenSSL을 사용하여 외부 키를 설정할 수 있도록 클러스터 또는 KMIP 서버에 SSL 인증서가 필요합니다.

단계

1. 키 공급자 클라이언트에 대한 인증서를 준비합니다. 클라이언트 인증서, 클라이언트 개인 키 및 신뢰 CA 번들을 포함합니다.



예를 들어, OpenSSL을 사용하여 외부 키를 설정할 수 있도록 클러스터 또는 KMIP 서버에 SSL 인증서를 준비합니다.

2. Astra Data Store 클러스터의 노드 중 하나에 로그인합니다.
3. 다음 kubectl extension 명령을 입력하여 Astra Data Store 클러스터의 주요 공급자를 구성합니다.

```
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>
```

다음 예에서는 ADS 클러스터 "astrads-cluster-f23d158"에 대해 "hashicorp"라는 외부 키 공급자를 구성합니다.

```
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. AstraDSCluster CR을 통해 sear에 외부 키 관리자를 사용하도록 Astra Data Store 클러스터를 구성합니다. 도움말을 표시합니다.

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

응답:

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
...
```

다음 명령을 실행하면 Astra Data Store 클러스터가 sear의 Key Manager로 "AstraDSKeyProvider hashicorp"를 사용하도록 구성됩니다. 이 명령은 또한 기본값 90일(2160시간)의 키 회전 시간을 사용합니다.

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



소프트웨어 암호화 유효 상태를 확인합니다

저장된 소프트웨어 암호화 구성을 확인할 수 있습니다.

단계

1. AstraDSCluster CR을 검사합니다.

```
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:              nnnnnnnnnn-nnnnn-nnnnn-nnnnn-nnnnnnnnnnnnnnn
```

내부 키 관리를 외부로 변경합니다

현재 외부 키 관리자를 사용 중인 경우 내부 키 관리자로 변경할 수 있습니다.

단계

1. SoftwareEncryptionAtRest 구성을 제거하여 AstraDSCluster CR을 변경합니다.
2. (선택 사항) 이전 AstraDSKeyProvider 및 관련 암호를 삭제합니다.



이전 키 공급자와 암호는 자동으로 제거되지 않습니다.

보안을 위해 키를 회전합니다

키 로테이션을 통해 보안이 강화됩니다. 기본적으로 Astra Data Store는 90일마다 자동으로 키를 순환합니다. 기본 설정을 변경할 수 있습니다. 또한 필요할 때 키를 회전할 수도 있습니다.

자동 키 회전을 구성합니다

1. CRD에서 AstraDSSEARKeyRotate 매개변수를 업데이트합니다.

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

주문형 키 회전을 구성합니다

1. 키를 회전하기 위해 AstraDSSEARKeyRotateRequest CR을 생성합니다.

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

## Astra Data Store 라이선스를 업데이트합니다

평가 기간을 연장하기 위해 Astra Data Store에 대해 설치된 평가 라이선스를 업데이트할 수 있습니다. 다음 세 가지 방법 중 하나를 사용하여 라이선스를 업데이트할 수 있습니다.

- Astra Control Center를 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 ["스토리지 백엔드 라이선스를 업데이트합니다"](#).
- Astra VMware 플러그인을 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 ["VMware를 사용하여 Astra Data Store를 관리합니다"](#).
- 명령줄을 사용하여 Astra Data Store 라이선스를 업데이트하려면 을 참조하십시오 [\[Update the Astra Data Store license using the command line\]](#).

### 명령줄을 사용하여 Astra Data Store 라이선스를 업데이트합니다

kubeck 유틸리티를 사용하여 Astra Data Store 라이선스를 업데이트할 수 있습니다.

단계

1. NetApp에서 구입한 대체 NetApp 라이선스 파일(NLF)을 적용합니다. 명령을 실행하기 전에 클러스터 이름(<<Astra-Data-Store-cluster-name>>)과 라이선스 파일 경로('<file\_path/file.txt>')를 입력합니다.

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

2. 라이선스가 추가되었는지 확인합니다.

```
kubectl astrads license list
```

다음과 유사한 응답이 표시됩니다.

NAME	ADSCLUSTER	VALID	PRODUCT
EVALUATION	ENDDATE	VALIDATED	
p1000000006	astrads-example-cluster	true	Astra Data Store
2023-01-23	2022-04-04T14:38:54Z		true

## Astra Data Store를 업그레이드합니다

Astra Data Store를 업그레이드하여 최신 기능과 수정 사항을 활용할 수 있습니다. Astra Data Store의 kubeck 확장자로 Astra Data Store를 업그레이드할 수 있습니다.

### kubectl을 사용하여 Astra Data Store를 업그레이드합니다

Astra Data Store의 kubectl을 이용하여 Astra Data Store를 업그레이드할 수 있습니다.

#### Astra Data Store 번들을 다운로드하고 이미지를 추출합니다

단계

1. 에 로그인합니다 "[NetApp Support 사이트](#)" Astra Data Store 번들('Astra\_Data\_Store\_2022.05.tar')을 다운로드하십시오.
2. (선택 사항) 다음 명령을 사용하여 번들의 서명을 확인합니다.

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

3. 디렉토리 생성:

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

4. 이미지 추출:

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



이미지는 작업 디렉토리 내에 생성된 'Astrads/images' 디렉토리에 압축이 풀립니다.

이진 파일을 복사하고 이미지를 로컬 레지스트리에 푸시합니다

단계

1. 쿠버네티스 쿠버네티스 바이너리가 설치된 표준 경로에 이미지를 추출하는 데 사용한 디렉토리에서 kubectl-astra 바이너리를 복사합니다("/usr/bin"은 아래 예에서 경로로 사용됨). kubectl-astra 데이터 저장소 클러스터를 설치 및 관리하는 사용자 지정 kubectl 확장입니다.



kubbeck 명령을 사용하여 kubectl 바이너리가 설치된 경로를 찾습니다.

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

2. Astra Data Store 이미지 디렉토리의 파일을 로컬 레지스트리에 추가합니다.



아래 이미지의 자동 로드에 대한 샘플 스크립트를 참조하십시오.

- a. 레지스트리에 로그인합니다.

```
docker login [your_registry_path]
```

- b. 환경 변수를 레지스트리 경로로 설정하여 Astra Data Store 이미지를 푸시합니다(예: REpo.company.com).

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

- c. 다음 스크립트를 실행하여 이미지를 Docker에 로드하고, 이미지에 태그를 지정한 다음 이미지를 로컬 레지스트리에 푸시합니다.

```
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
```

업그레이드를 수행합니다

단계

1. Astradsoperator.YAML 파일을 로컬 디렉토리에 복사합니다.

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

2. 운영자를 업그레이드 한다. 대문자의 인수를 환경에 적합한 정보로 바꿉니다.

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

3. Astra Data Store 업그레이드를 시작합니다. 대문자의 인수를 환경에 적합한 정보로 바꿉니다.

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

업그레이드가 시작되었다는 메시지가 나타나고 완료하는 데 몇 분 정도 걸립니다.

## 자동 스크립트로 Astra Data Store를 제거합니다

Astra Data Store와 제어 플레인을 제거하려면 워크로드, 바인딩, 볼륨, 내보내기 정책, Astra Data Store 클러스터, 라이선스, 배포 환경 및 Astra Data Store 네임스페이스를 제거합니다.

설치 제거 방법은 다음과 같습니다.

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

### 자동 스크립트로 Astra Data Store를 제거합니다

이 프로세스는 자동 스크립트를 사용하여 Astra Data Store를 제거합니다.

무엇을 '필요로 할거야

- 루트 관리 권한

Astra Data Store 제거 프로세스는 다음과 같은 고급 단계를 안내합니다.

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

기존 워크로드 및 바인딩을 제거합니다

Astra Data Store를 제거하기 전에 먼저 다음을 제거해야 합니다

- Astra Data Store를 스토리지 백엔드로 사용하는 모든 애플리케이션 워크로드
- Astra Data Store를 백엔드로 사용하는 Trident 바인딩

이렇게 하면 Kubernetes 환경이 클린 상태로 유지됩니다. 이는 다시 설치할 때 중요합니다.

### Astra Data Store 클러스터를 제거합니다

Astra Data Store를 설치 제거하려면 NetApp Support 사이트에서 다운로드한 Astra Data Store tar 파일에 있는 `uninstall.sh` 스크립트를 사용하십시오.

1. 'manifests' 디렉토리에서 `uninstall.sh`를 찾습니다.
2. 다음 'ed' 명령을 실행합니다.

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

3. 제거할 항목을 나타내는 다음 스크립트를 실행합니다.

```
./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. 클러스터를 설치 제거하려면 "`uninstall.sh <cluster>`"를 입력하십시오

그렇지 않으면 모든 항목을 제거하려면 `uninstall.sh`를 입력합니다



대부분의 경우 모든 항목을 제거합니다. 클러스터를 나중에 다시 배포하려는 경우 클러스터만 제거하면 됩니다.

5. 프롬프트에서 계속 진행할지 확인하고 `erasedata`를 입력합니다

응답:

```
./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                    |
+-----+
```

### Astrads-system namespace의 제거를 검증합니다

다음 명령을 실행하면 결과가 반환되지 않는지 확인합니다.

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

컨테이너를 작업자 노드에서 실행하지 않도록 합니다

소방관이나 netwd 같은 컨테이너가 작업자 노드에서 실행되고 있지 않는지 확인합니다. 각 노드에서 다음을 실행합니다.

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

### OpenShift Container Platform 리소스를 삭제합니다

Red Hat OpenShift Container Platform(OCP)에 Astra Data Store를 설치한 경우 OCP SCC(Security Context Constraints) 및 rolebindings 리소스를 제거할 수 있습니다.

OpenShift는 POD가 수행할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 사용합니다.

표준 제거 프로세스를 완료한 후 다음 단계를 완료합니다.

1. SCC 리소스 제거:

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

## 2. rolebindings 리소스 제거:

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



이 단계에서는 "리소스를 찾을 수 없음" 오류를 무시합니다.

## 스크립트 없이 **Astra Data Store**를 수동으로 제거합니다

이 프로세스는 스크립트 없이 Astra Data Store를 수동으로 제거합니다.

자동 스크립트 없이 Astra Data Store를 수동으로 제거하려면 워크로드, 바인딩, 볼륨, 익스포트 정책, 클러스터, 라이선스, 배포 환경 및 Astra Data Store 네임스페이스.

무엇을 '필요로 할거야

- 루트 관리 권한

Astra Data Store 제거 프로세스는 다음과 같은 고급 단계를 안내합니다.

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

기존 워크로드 및 바인딩을 제거합니다

Astra Data Store를 제거하기 전에 먼저 다음을 제거해야 합니다

- Astra Data Store를 스토리지 백엔드로 사용하는 모든 애플리케이션 워크로드
- Astra Data Store를 백엔드로 사용하는 Trident 바인딩

이렇게 하면 Kubernetes 환경이 클린 상태로 유지됩니다. 이는 다시 설치할 때 중요합니다.

## **Astra Data Store** 클러스터와 컨트롤 플레인을 제거합니다

Astra Data Store를 수동으로 제거하려면 아래 단계를 따르십시오.

볼륨 및 익스포트 정책을 삭제합니다

클러스터를 삭제하기 전에 Astra Data Store 볼륨과 익스포트 정책을 삭제해야 합니다.



먼저 볼륨 및 익스포트 정책을 삭제하지 않으면 Astra Data Store 볼륨 객체가 삭제될 때까지 클러스터 삭제 프로세스가 일시 중지됩니다. 클러스터 삭제를 시작하기 전에 이러한 항목을 제거하는 것이 더 효율적입니다.

## 단계

### 1. 볼륨 삭제:

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

### 2. 익스포트 정책 삭제:

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

## Astra Data Store 클러스터를 삭제합니다

클러스터를 삭제하면 클러스터 범위 리소스와 함께 Astra CR(Data Store Cluster Object Custom Resource)만 삭제됩니다.



클러스터가 삭제된 후에도 운영자, 노드 정보 포드 및 클러스터 컨트롤러(Kubernetes 범위 리소스)는 그대로 유지됩니다.

클러스터를 삭제하면 기본 운영 체제가 노드에서 제거되므로 'firtap' 및 'netwd' 서비스가 중지됩니다.

제거 프로그램을 완료하는 데 약 1분 정도 걸립니다. 그런 다음 Astra Data Store 클러스터 범위 리소스 제거가 시작됩니다.

### 1. 클러스터를 삭제합니다.

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

## 라이센스를 삭제합니다

1. 클러스터의 각 작업자 노드에 SSH를 통해 'firewTap' 또는 'netwd'가 작업자 노드에서 실행되고 있지 않음을 확인합니다.
2. Astra Data Store 라이선스를 삭제합니다.

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

## Astra Data Store 설치를 삭제합니다

클러스터에서 컨트롤러, 운영자, 네임스페이스 및 지원 포드를 삭제하십시오.

1. Astra Data Store 설치 객체를 삭제합니다.

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

2. 데이터 저장소 DemonSets 및 모든 Astra Data Store 컨트롤러 리소스를 삭제합니다.

```
~% 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. 나머지 아티팩트 및 작업자 YAML 파일 삭제:

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

## Astrads-system namespace의 제거를 검증합니다

다음 명령을 실행하면 결과가 반환되지 않는지 확인합니다.

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

컨테이너를 작업자 노드에서 실행하지 않도록 합니다

소방관이나 netwd 같은 컨테이너가 작업자 노드에서 실행되고 있지 않는지 확인합니다. 각 노드에서 다음을 실행합니다.

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

## OpenShift Container Platform 리소스를 삭제합니다

Red Hat OpenShift Container Platform(OCP)에 Astra Data Store를 설치한 경우 OCP SCC(Security Context Constraints) 및 rolebindings 리소스를 제거할 수 있습니다.

OpenShift는 POD가 수행할 수 있는 작업을 제어하는 SCC(Security Context Constraints)를 사용합니다.

표준 제거 프로세스를 완료한 후 다음 단계를 완료합니다.

1. SCC 리소스 제거:

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

2. rolebindings 리소스 제거:

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



이 단계에서는 "리소스를 찾을 수 없는 오류"를 무시합니다.

수동 삭제 샘플

다음은 실행 수동 제거 스크립트의 예입니다.

```
$ 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

```

## Astra Data Store 제거 프로세스 문제를 해결합니다

제거 프로세스 문제를 해결해야 하는 경우 다음 권장 사항을 검토하십시오.

### 종료 상태의 포드

Astra Data Store 제거 프로세스를 사용하면 Kubernetes에서 POD가 종료 상태로 유지되는 경우가 있습니다.



이 문제가 발생하면 다음 명령을 실행하여 "astrads-system" 네임스페이스의 모든 Pod를 강제로 삭제합니다.

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

서비스 품질 정책은 기존 클러스터를 가리킵니다

Astra Data Store Cluster만 삭제하고 다시 배포하는 경우, QoS(Quality of Service) 정책이 이전 클러스터를 가리키므로 영구 볼륨 클레임(PVC) 또는 볼륨을 생성하지 못할 수 있습니다.

1. 이를 방지하려면 Astra Data Store 클러스터를 삭제한 후 QoS 정책을 수동으로 삭제하십시오.

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

2. 전체 Astra Data Store 구축(클러스터뿐 아니라)을 삭제합니다.

```
uninstall.sh all
```

**Astra Data Store**를 삭제 또는 제거한 후 키 제공자 **CRS**가 제거되지 않았습니다

삭제 또는 제거 중인 Astra Data Store 클러스터에 대해 외부 키 공급자가 구성된 경우 제거되지 않은 키 공급자 CR을 수동으로 정리해야 할 수 있습니다.

## 예 1. 세부 정보

다음 해결 방법을 사용합니다.

단계

1. 키 제공자 CRS가 제거되지 않았는지 확인합니다.

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

응답:

NAME	AGE
externalkeyprovider1	94s

2. 키 공급자 CRS를 제거합니다.

- a. 종료자를 제거합니다.

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

- b. 아래에 강조 표시된 종료자 라인을 제거합니다.

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

```

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

```

c. 종료자를 제거한 후 키 공급자 CR을 삭제합니다.

```

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

```

## 저작권 정보

Copyright © 2022 NetApp, Inc. All rights reserved. 미국에서 인쇄된 본 문서의 어떤 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 그래픽, 전자적 또는 기계적 수단(사진 복사, 레코딩 등)으로도 저작권 소유자의 사전 서면 승인 없이 전자 검색 시스템에 저장 또는 저장.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지 사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 "있는 그대로" 제공되며 상품성 및 특정 목적에 대한 적합성에 대한 명시적 또는 묵시적 보증을 포함하여 이에 제한되지 않고, 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 또는 파생적 손해(소계 물품 또는 서비스의 조달, 사용 손실, 데이터 또는 수익 손실, 계약, 엄격한 책임 또는 불법 행위(과실 또는 그렇지 않은 경우)에 관계없이 어떠한 책임도 지지 않으며, 이는 이러한 손해의 가능성을 사전에 알고 있던 경우에도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구입의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허 또는 해외 특허, 해외 특허, 해외 특허, 해외 특허, 해외 특허, 해외 특허, 해외 특허, 해외 특허, 미국 출원 중인 특허로 보호됩니다.

권리 제한 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.277-7103(1988년 10월) 및 FAR 52-227-19(1987년 6월)의 기술 데이터 및 컴퓨터 소프트웨어의 권리(Rights in Technical Data and Computer Software) 조항의 하위 조항 (c)(1)(ii)에 설명된 제한사항이 적용됩니다.

## 상표 정보

NETAPP, NETAPP 로고 및 에 나열된 마크는 NetApp에 있습니다 <http://www.netapp.com/TM> 는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.