安装概述
Astra Data Store

NetApp
January 27, 2022

目录

Astra 数据存储安装概述

选择并完成以下 Astra Data Store 安装过程之一：

- "使用标准流程安装 Astra 数据存储"。
- "如果您使用的是 Red Hat OpenShift ，请使用 OpenShift 安装 Astra Data Store"。

安装 Astra Data Store 预览版

要安装 Astra Data Store 预览版，请从下载安装包 "NetApp 支持站点" 并完成此操作步骤中所述的安装步骤。

或者，您也可以 "在 Red Hat OpenShift 容器平台（ OCP ）上安装 Astra Data Store 预览版"。

您需要什么？ **#8217** ；将需要什么

- "开始安装之前，请为 Astra Data Store 预览部署准备您的环境"。
- 访问 "NetApp 支持站点"。 如果您还没有完全访问的 NetApp 支持站点帐户，请进行预览。
- 答 "NetApp 许可证文件（ NLF ）" 用于 Astra Data Store 预览。下载许可证的说明将在您之后发送给您 "注册"。
- 具有活动上下文集群管理员权限的活动 kubeconfig 。
- 了解 "角色和权限" 由 Astra Data Store 预览版使用。
- Internet 连接。Astra Data Store 预览版不支持带气流的环境。要直接或通过代理访问 support.netapp.com ，需要 Internet 连接。

Astra Data Store 预览安装过程将指导您完成以下高级步骤：

- [Download the Astra Data Store preview bundle and extract the images]
- [Copy the binary and push images to your local registry]
- [Install the Astra Data Store preview operator]
- [Deploy the Astra Data Store preview version YAML]
- [Apply the Astra Data Store preview license]
- [Install the Astra Data Store preview cluster]
- [Understand deployment-related events]
- [Configure Astra Data Store preview monitoring]

下载 Astra Data Store 预览包并提取映像

1. 登录到 "NetApp 支持站点" 并下载 Astra Data Store 预览包（`2021.12_asadatastore.tar`）。
2. （可选）使用以下命令验证捆绑包的签名：

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

3. 提取映像：

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

将二进制文件和推送映像复制到本地注册表

1. 将 kubectl-astrad 二进制文件从用于提取映像的目录复制到安装了 K8s kubectl 二进制文件的标准路径；例如，`/usr/bin/`。kubectl-astrad 是一个自定义 kubectl 扩展，用于安装和管理 Astra Data Store 预览集群。

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

2. 将 Astra Data Store 预览映像目录中的文件添加到本地注册表中。

   有关自动加载映像的信息，请参见下面的示例脚本。

   a. 登录到注册表：

```
docker login [your_registry_path]
```

   b. 将环境变量设置为要推送 Astra Data Store 预览映像的注册表路径；例如， `repo.company.com`。

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

c. 运行脚本将映像加载到 Docker ，标记映像，并将这些映像推送到本地注册表：

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

安装 **Astra Data Store** 预览运算符

1. 列出 Astra Data Store 预览清单：

```
ls manifests/*yaml
```

响应：

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

2. 使用 kubectl apply 部署操作员：

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

响应：

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
```

```
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.netapp.io created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
```

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

3. 验证 Astra 数据存储操作员 POD 是否已启动且正在运行：

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

响应：

```
NAME                                READY   STATUS    RESTARTS   AGE
astrads-operator-5ffb94fbf-7ln4h    1/1     Running   0          17m
```

部署 **Astra Data Store** 预览版 **YAML**

1. 使用 kubectl Deploy 应用：

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

2. 验证 Pod 是否正在运行：

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

响应：

```
NAME                                          READY   STATUS    RESTARTS
AGE
astrads-cluster-controller-7f6f884645-xxf2n   1/1     Running   0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv      1/1     Running   0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j    1/1     Running   0
2m2s
astrads-operator-5ffb94fbf-7ln4h              1/1     Running   0
2m10s
```

应用 Astra Data Store 预览许可证

1. 应用在注册预览时获取的 NetApp 许可证文件（ NLF ）。运行命令之前，请输入您所在集群的名称（` <Astra-Data-Store-cluster-name>` ） 即将部署 或已部署许可证文件的路径（` <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. 验证是否已添加此许可证：

```
kubectl astrads license list
```

响应：

```
NAME        ADSCLUSTER                VALID   PRODUCT
EVALUATION  ENDDATE     VALIDATED
p100000006  astrads-example-cluster    true    Astra Data Store Preview
true        2022-01-23  2021-11-04T14:38:54Z
```

安装 Astra Data Store 预览集群

1. 打开 YAML 文件：

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

2. 编辑 YAML 文件中的以下值。

   以下步骤将提供一个简化的 YAML 文件示例。

   a. （必需） * 元数据 * ： 在 `metadata` 中，将 `name` string 更改为集群名称。此集群名称必须与您在使用时使用的集群名称相同 应用许可证。

   b. （必需） * 规格 * ： 在 `sPec` 中更改以下必需值：

      - 将 `mVIP` 字符串更改为可从集群中的任何工作节点路由的浮动管理 IP 的 IP 地址。
      - 在 `adsDataNetworks` 中，添加一个逗号分隔的浮动 IP 地址列表（`addresses` ），这些地址可从要挂载 NetApp 卷的任何主机路由。每个节点使用一个浮动 IP 地址。数据网络 IP 地址的数量应至少与 Astra Data Store 预览节点的数量相同。对于 Astra Data Store 预览版，这意味着至少有 4 个地址，如果您计划稍后将集群扩展到 5 个节点，则至少需要 5 个地址。
      - 在 `adsDataNetworks` 中，指定数据网络使用的网络掩码。
      - 在 `adsNetworkInterfaces` 中，将 `mGMT` 和 `data` 值替换为要用于管理，集群和存储的网络接口名称。如果未指定名称，则节点的主接口将用于管理，集群和存储网络连接。

        集群和存储网络必须位于同一接口上。Astra Data Store 预览管理界面应与 Kubernetes 节点的管理界面相同。

   c. （可选） * 显示器配置 * ： 如果要配置 监控操作员 （如果您不使用 Astra Control Center 进行监控，则可选），从部分中删除注释，添加应用代理 CR （监控操作员资源）的命名空间（默认值为 `netapp-monitoring` ），然后添加您在先前步骤中使用的注册表的 repo路径 （`yor_registry_path` ）。

   d. （可选） * 自动支持配置 * ： 保留 "AutoSupport" 默认值，除非您需要配置代理：

      - 对于 `proxyURL` ，使用要用于 AutoSupport 捆绑包传输的端口设置代理的 URL 。

      大多数注释已从以下 YAML 示例中删除。

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 34
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
```

```
  # Specify the network interface names to use for management, cluster
and storage networks.
  # If none are specified, the node's primary interface will be used for
management, cluster and storage networking.
  # To move the cluster and storage networks a different interface than
management, specify all three interfaces to use here.
  # The Astra Data Store management interface should be same as the
Kubernetes node’s management interface.
  # NOTE: The cluster and storage networks need to be on the same
interface.
  adsNetworkInterfaces:
    managementInterface: "mgmt"
    clusterInterface: "data"
    storageInterface: "data"
  # [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
 # monitoringConfig:
   # namespace: "netapp-monitoring"
   # repo: "[YOUR REGISTRY]"
  autoSupportConfig:
    autoUpload: true
    enabled: true
    coredumpUpload: false
    historyRetentionCount: 25
    destinationURL: "https://support.netapp.com/put/AsupPut"
    # ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
    # proxyURL:
    periodic:
      - schedule: "0 0 * * *"
        periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
          userMessage: Daily Monitoring Storage AutoSupport bundle
          nodes: all
        - component:
            name: controlplane
            event: daily
          userMessage: Daily Control Plane AutoSupport bundle
```

3. 使用 `kubectl apply` 部署集群：

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

4. 等待几分钟，以完成集群创建操作，然后验证 Pod 是否正在运行：

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

响应示例：

```
NAME                        READY     STATUS    RESTARTS    AGE
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-1ab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

5. 验证集群部署进度：

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

响应示例：

```
NAME                          STATUS    VERSION    SERIAL NUMBER    MVIP
AGE

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

了解与部署相关的事件

在集群部署期间，操作状态应从 `blank` 更改为 `in progress` 更改为 `created` 。集群部署将持续大约 8 到 10 分钟。要在部署期间监控集群事件，您可以运行以下命令之一：

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n
astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

以下是部署期间的关键事件：

事件消息	含义
已成功选择 4 个控制面板节点加入 ADS 集群	Astra Data Store 预览运算符可确定具有 CPU ，内存，存储和网络连接的节点足以创建 Astra Data Store 预览集群。
ADS 集群创建正在进行中	Astra Data Store 预览集群控制器已启动集群创建操作。
已创建 ADS 集群	已成功创建集群。

如果集群的状态未更改为 `in progress` ，请查看操作员日志，了解有关节点选择的更多详细信息：

```
kubectl logs -n astrads-system <astrads operator pod name>
```

如果集群状态停留在 `in progress` ，请检查集群控制器的日志：

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

配置 **Astra Data Store** 预览监控

您可以为 Astra 控制中心监控或其他遥测服务监控配置 Astra 数据存储预览。

为 **Astra** 控制中心预览配置监控

只有在 Astra 控制中心将 Astra Data Store 预览作为后端进行管理后，才能执行以下步骤。

1. 配置 Astra 数据存储预览以供 Astra 控制中心监控：

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

安装监控操作员

（可选）只有当 Astra Data Store 预览版不会导入到 Astra 控制中心中时，才建议使用监控操作员。如果您的 Astra 数据存储预览实例是独立部署，使用 Cloud Insights 监控遥测或将日志流式传输到 Elastic 等第三方端点，则可以安装监控操作员。

1. 运行此安装命令：

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

2. 配置用于监控的 Astra Data Store 预览版：

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

下一步行动

执行以完成部署 "设置任务"。

在 Red Hat OpenShift 容器平台上安装 Astra Data Store 预览版

要在 Red Hat OpenShift 容器平台（ OCP ）上安装 Astra Data Store preview ，请从下载安装包 "NetApp 支持站点" 并完成此操作步骤中所述的安装步骤。

您需要什么？ **#8217** ；将需要什么

- 开始安装之前， "为部署 Astra Data Store 准备您的环境"。
- 访问 "NetApp 支持站点"。 如果您还没有完全访问的 NetApp 支持站点帐户，请进行预览。
- 答 "NetApp 许可证文件" （ nlf ），用于 Astra Data Store 预览。注册后，系统将向您发送许可证下载说明。
- 具有活动上下文集群管理员权限的活动 kubeconfig 。
- 了解 "角色和特权" 由 Astra Data Store 预览版使用。
- Internet 连接。Astra Data Store 预览版不支持带气流的环境。要直接或通过代理访问 support.netapp.com ，需要 Internet 连接。

关于此任务

Astra Data Store 预览安装过程将指导您完成以下高级步骤：

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

- [Understand deployment-related events]
- [Configure Astra Data Store preview monitoring]
- [Install the monitoring operator]
- [Set up iptables]

下载 Astra Data Store 预览包并提取映像

1. 登录到 "NetApp 支持站点" 并下载 Astra Data Store 预览包（`2021.12_asadatastore.tar`）。
2. （可选）验证捆绑包的签名：

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

3. 提取映像：

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

将二进制文件和推送映像复制到本地注册表

1. 将 kubectl-astrad 二进制文件从用于提取映像的目录复制到安装了 K8s kubectl 二进制文件的标准路径；例如，`/usr/bin/`。kubectl-astrad 是一个自定义 kubectl 扩展，用于安装和管理 Astra Data Store 预览集群。

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

2. 将 Astra Data Store 预览映像目录中的文件添加到本地注册表中。

   有关自动加载映像的信息，请参见下面的示例脚本。

   a. 登录到注册表：

```
docker login [your_registry_path]
```

   b. 将环境变量设置为要推送 Astra Data Store 预览映像的注册表路径；例如，`repo.company.com`。

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

   c. 运行脚本将映像加载到 Docker ，标记映像，并将这些映像推送到本地注册表：

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

创建命名空间以部署 **Astra Data Store** 预览版

创建一个 `astrads-system` 命名空间，其中将安装所有 Astra Data Store 预览组件。

1. 创建命名空间：

```
kubectl create -f ads_namespace.yaml
```

示例： ads_namespace.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: operator
  name: astrads-system
```

创建自定义 **SCC**

OpenShift 使用安全上下文约束（ SCC ）来控制 Pod 可以执行的操作。默认情况下，任何容器的执行都将获得受限的 SCC ，并且仅获得该 SCC 定义的功能。

受限 SCC 不提供 Astra Data Store 预览集群 Pod 所需的权限。使用此操作步骤可为 Astra 数据存储预览提供所需的权限（在示例中列出）。

将自定义 SCC 分配给 Astra Data Store 预览命名空间的默认服务帐户。

步骤

1. 创建自定义 SCC：

```
kubectl create -f ads_privileged_scc.yaml
```

示例： ads_privileged_scc.yaml

```
allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegeEscalation: true
allowPrivilegedContainer: true
allowedCapabilities:
- '*'
allowedUnsafeSysctls:
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: null
fsGroup:
  type: RunAsAny
groups: []
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'ADS privileged. Grant with caution.'
    release.openshift.io/create-only: "true"
  name: ads-privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: null
runAsUser:
  type: RunAsAny
seLinuxContext:
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups:
  type: RunAsAny
users:
- system:serviceaccount:astrads-system:default
volumes:
- '*'
```

2. 使用 `oc get SCC` 命令显示新添加的 SCC：

```
# oc get scc/ads-privileged
NAME              PRIV    CAPS    SELINUX    RUNASUSER   FSGROUP
SUPGROUP   PRIORITY     READONLYROOTFS   VOLUMES
ads-privileged   true    ["*"]   RunAsAny   RunAsAny    RunAsAny
RunAsAny   <no value>   false            ["*"]
#
```

创建角色和角色绑定

为 Astra Data Store 预览版创建所需的角色和角色绑定，以供默认服务帐户使用。

以下 YAML 定义可分配 `astrads.netapp.io` API 组中的 Astra Data Store 预览资源所需的各种角色（通过绑定）。

1. 创建定义的角色和角色绑定：

```
kubectl create -f oc_role_bindings.yaml
```

示例： oc_role_Bindings.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: privcrole
rules:
- apiGroups:
  - security.openshift.io
  resourceNames:
  - ads-privileged
  resources:
  - securitycontextconstraints
  verbs:
  - use
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: default-scc-rolebinding
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: privcrole
subjects:
```

```
- kind: ServiceAccount
  name: default
  namespace: astrads-system
---

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ownerref
  namespace: astrads-system
rules:
- apiGroups:
  - astrads.netapp.io
  resources:
  - '*/finalizers'
  verbs:
  - update
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: or-rb
  namespace: astrads-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: ownerref
subjects:
- kind: ServiceAccount
  name: default
  namespace: astrads-system
```

准备工作节点

准备用于 Astra Data Store 预览集群部署的工作节点。在 Astra 数据存储预览集群使用的所有工作节点上执行此操作步骤。

OpenShift 对 kubelet 配置文件（` /var/lib/kubelet/config.json` ）使用 json 格式。Astra Data Store 预览集群会查找 `kubelet config` 文件的 YAML 格式。

步骤

1. 在启动集群安装之前，在每个工作节点上创建 ` /var/lib/kubelet/config.yaml` 文件。

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

2. 在应用集群 YAML 之前，在所有 Kubernetes 节点上完成此操作步骤。

如果不执行此操作，则 Astra Data Store 预览集群安装将失败。

安装 Astra Data Store 预览运算符

1. 列出 Astra Data Store 预览清单：

```
ls manifests/*yaml
```

响应：

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

2. 使用 `kubectl apply` 命令部署操作员：

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

响应：

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsautosupports.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.ne
tapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnfsoptions.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsnodeinfoes.astrads.
netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
```

```
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumefiles.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumes.astrads.net
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.ast
rads.netapp.io created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscloudsnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradscluster-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradslicense-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolume-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-autosupport-viewer-role
created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-metrics-reader created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappexportpolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsdeployment-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-netappsdsnodeinfo-viewer-
role created
```

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

3. 验证 Astra 数据存储操作员 POD 是否已启动且正在运行：

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

响应：

```
NAME                                  READY   STATUS    RESTARTS   AGE
astrads-operator-5ffb94fbf-7ln4h      1/1     Running   0          17m
```

部署 Astra Data Store 预览版 YAML

1. 使用 `kubectl apply` 命令进行部署：

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

2. 验证 Pod 是否正在运行：

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

响应：

```
NAME                                          READY   STATUS    RESTARTS
AGE
astrads-cluster-controller-7f6f884645-xxf2n   1/1     Running   0
117s
astrads-ds-nodeinfo-astradsversion-2jqnk      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-dbk7v      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-rn9tt      1/1     Running   0
2m7s
astrads-ds-nodeinfo-astradsversion-vsmhv      1/1     Running   0
2m7s
astrads-license-controller-fb8fd56bc-bxq7j    1/1     Running   0
2m2s
astrads-operator-5ffb94fbf-7ln4h              1/1     Running   0
2m10s
```

应用 **Astra Data Store** 预览许可证

1. 应用在注册预览时获取的 NetApp 许可证文件（ NLF ）。运行命令之前，请输入您所在集群的名称（` <Astra-Data-Store-cluster-name>` ） 即将部署 或已部署许可证文件的路径（` <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. 验证是否已添加此许可证：

```
kubectl astrads license list
```

响应：

```
NAME        ADSCLUSTER                  VALID   PRODUCT
EVALUATION  ENDDATE     VALIDATED
p100000006  astrads-example-cluster    true    Astra Data Store Preview
true        2022-01-23  2021-11-04T14:38:54Z
```

安装 **Astra Data Store** 预览集群

1. 打开 YAML 文件：

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

2. 编辑 YAML 文件中的以下值。

   以下步骤将提供一个简化的 YAML 文件示例。

   a. （必需） * 元数据 * ： 在 `metadata` 中，将 `name` string 更改为集群名称。此集群名称必须与您在使用时使用的集群名称相同 应用许可证。

   b. （必需） * 规格 * ： 在 `sPec` 中更改以下必需值：

      - 将 `mVIP` 字符串更改为可从集群中的任何工作节点路由的浮动管理 IP 的 IP 地址。
      - 在 `adsDataNetworks` 中，添加一个逗号分隔的浮动 IP 地址列表（`addresses` ），这些地址可从要挂载 NetApp 卷的任何主机路由。每个节点使用一个浮动 IP 地址。数据网络 IP 地址的数量应至少与 Astra Data Store 预览节点的数量相同。对于 Astra Data Store 预览版，这意味着至少有 4 个地址，如果您计划稍后将集群扩展到 5 个节点，则至少需要 5 个地址。
      - 在 `adsDataNetworks` 中，指定数据网络使用的网络掩码。
      - 在 `adsNetworkInterfaces` 中，将 `mGMT` 和 `data` 值替换为要用于管理，集群和存储的网络接口名称。如果未指定名称，则节点的主接口将用于管理，集群和存储网络连接。

        集群和存储网络必须位于同一接口上。Astra Data Store 预览管理界面应与 Kubernetes 节点的管理界面相同。

   c. （可选） * 显示器配置 * ： 如果要配置 监控操作员 （如果您不使用 Astra Control Center 进行监控，则可选），从部分中删除注释，添加应用代理 CR （监控操作员资源）的命名空间（默认值为 `netapp-monitoring` ），然后添加您在先前步骤中使用的注册表的 repo路径 （`yor_registry_path` ）。

   d. （可选） * 自动支持配置 * ： 保留 "AutoSupport" 默认值，除非您需要配置代理：

      - 对于 `proxyURL` ，使用要用于 AutoSupport 捆绑包传输的端口设置代理的 URL 。

      大多数注释已从以下 YAML 示例中删除。

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 34
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
```

```
  # Specify the network interface names to use for management, cluster
and storage networks.
  # If none are specified, the node's primary interface will be used for
management, cluster and storage networking.
  # To move the cluster and storage networks a different interface than
management, specify all three interfaces to use here.
  # The Astra Data Store management interface should be same as the
Kubernetes node’s management interface.
  # NOTE: The cluster and storage networks need to be on the same
interface.
  adsNetworkInterfaces:
    managementInterface: "mgmt"
    clusterInterface: "data"
    storageInterface: "data"
  # [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
 # monitoringConfig:
   # namespace: "netapp-monitoring"
   # repo: "[YOUR REGISTRY]"
  autoSupportConfig:
    autoUpload: true
    enabled: true
    coredumpUpload: false
    historyRetentionCount: 25
    destinationURL: "https://support.netapp.com/put/AsupPut"
    # ProxyURL defines the URL of the proxy with port to be used for
AutoSupport bundle transfer
    # proxyURL:
    periodic:
      - schedule: "0 0 * * *"
        periodicconfig:
        - component:
            name: storage
            event: dailyMonitoring
          userMessage: Daily Monitoring Storage AutoSupport bundle
          nodes: all
        - component:
            name: controlplane
            event: daily
          userMessage: Daily Control Plane AutoSupport bundle
```

3. 使用 `kubectl apply` 部署集群：

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

4. 如果已启用 SELinux ，请为 Astra Data Store 预览集群中节点上的以下目录重新标记 `selinux` 上下文。

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

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

> 之所以需要执行此步骤，是因为 `selinux` 会阻止这些目录处于可写状态，从而导致支持 Pod 进入 `CrashLoopBackoff` 状态。需要对 Astra Data Store 预览集群中的所有节点执行此步骤。

5. 等待几分钟，以完成集群创建操作，然后验证 Pod 是否正在运行：

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

响应示例：

```
NAME READY STATUS RESTARTS AGE
astrads-cluster-controller-7c67cc7f7b-2jww2 1/1 Running 0 7h31m
astrads-deployment-support-788b859c65-2qjkn 3/3 Running 19 12d
astrads-ds-astrads-cluster-1ab0dbc-j9jzc 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-k9wp8 1/1 Running 0 5d1h
astrads-ds-astrads-cluster-1ab0dbc-pwk42 1/1 Running 0 5d2h
astrads-ds-astrads-cluster-1ab0dbc-qhvc6 1/1 Running 0 8h
astrads-ds-nodeinfo-astradsversion-gcmj8 1/1 Running 1 12d
astrads-ds-nodeinfo-astradsversion-j826x 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-vdthh 1/1 Running 3 12d
astrads-ds-nodeinfo-astradsversion-xwgsf 1/1 Running 0 12d
astrads-ds-support-828vw 2/2 Running 2 5d2h
astrads-ds-support-cfzts 2/2 Running 0 8h
astrads-ds-support-nzkkr 2/2 Running 15 7h49m
astrads-ds-support-xxbnp 2/2 Running 1 5d2h
astrads-license-controller-86c69f76bb-s6fb7 1/1 Running 0 8h
astrads-operator-79ff8fbb6d-vpz9m 1/1 Running 0 8h
```

6. 验证集群部署进度：

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

响应示例：

```
NAME                          STATUS    VERSION    SERIAL NUMBER    MVIP
AGE

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

了解与部署相关的事件

在集群部署期间，操作状态应从 `blank` 更改为 `in progress` 更改为 `created` 。集群部署将持续大约 8 到 10 分钟。要在部署期间监控集群事件，您可以运行以下命令之一：

```
kubectl get events --field-selector involvedObject.kind=AstraDSCluster -n
astrads-system
```

```
kubectl describe astradscluster <cluster name> -n astrads-system
```

以下是部署期间的关键事件：

事件消息	含义
已成功选择 4 个控制面板节点加入 ADS 集群	Astra Data Store 预览运算符可确定具有 CPU ，内存，存储和网络连接的节点足以创建 Astra Data Store 预览集群。
ADS 集群创建正在进行中	Astra Data Store 预览集群控制器已启动集群创建操作。
已创建 ADS 集群	已成功创建集群。

如果集群的状态未更改为 `in progress` ，请查看操作员日志，了解有关节点选择的更多详细信息：

```
kubectl logs -n astrads-system <astrads operator pod name>
```

如果集群状态停留在 `in progress` ，请检查集群控制器的日志：

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

配置 **Astra Data Store** 预览监控

您可以为 Astra 控制中心监控或其他遥测服务监控配置 Astra 数据存储预览。

为 **Astra** 控制中心预览配置监控

只有在 Astra 控制中心将 Astra Data Store 预览作为后端进行管理后，才能执行以下步骤。

1. 配置 Astra 数据存储预览以供 Astra 控制中心监控：

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

安装监控操作员

（可选）只有当 Astra Data Store 预览版不会导入到 Astra 控制中心中时，才建议使用监控操作员。如果您的 Astra 数据存储预览实例是独立部署，使用 Cloud Insights 监控遥测或将日志流式传输到 Elastic 等第三方端点，则可以安装监控操作员。

1. 运行此安装命令：

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

2. 配置用于监控的 Astra Data Store 预览版：

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

设置 IP 表

OpenShift SDN （ OpenShift 的默认 CNI 插件）不会以掩码形式查询从主机环回接口（ 127.0.1/localhost ）到 HostPorts 的流量。

Astra Data Store 预览集群需要添加一个 NAT 规则，用于将流量从本地主机转发到集群服务端口（ 9920 ）。

步骤

1. 记下 NAT 表输出链中当前 `KUBE-HOSTPORTS` 规则的行号。

   在以下示例中， `KUBE-HOSTPORTS` 位于位置 4 。

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals
*/
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE:
this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type
LOCAL /* handle service NodePorts; NOTE: this must be the last rule in
the chain */
4 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */
ADDRTYPE match dst-type LOCAL
```

2. 向 `KUBE-HOSTPORTS` 条目上方的 NAT 表添加新规则。

```
$ sudo iptables -t nat -I OUTPUT 4 -s 127.0.0.1 -j KUBE-MARK-MASQ -p tcp
--dport 9920
```

3. 验证新添加的规则是否已添加到 `KUBE-HOSTPORTS` 规则上方的 NAT 表中。

```
$ sudo iptables -t nat -L OUTPUT --line-numbers
Chain OUTPUT (policy ACCEPT)
num target prot opt source destination
1 KUBE-SERVICES all -- anywhere anywhere /* kubernetes service portals
*/
2 KUBE-PORTALS-HOST all -- anywhere anywhere /* handle ClusterIPs; NOTE:
this must be before the NodePort rules */
3 KUBE-NODEPORT-HOST all -- anywhere anywhere ADDRTYPE match dst-type
LOCAL /* handle service NodePorts; NOTE: this must be the last rule in
the chain */
4 KUBE-MARK-MASQ tcp -- localhost anywhere tcp dpt:9920
5 KUBE-HOSTPORTS all -- anywhere anywhere /* kube hostport portals */
ADDRTYPE match dst-type LOCAL
```

在 Astra Data Store 预览集群中的所有节点上执行这些步骤。

如果 Astra Data Store 预览集群节点重新启动，请重复上述步骤。

下一步行动

执行以完成部署 "设置任务"。