入门

Astra Data Store

NetApp
January 27, 2022

目录

入门

Astra Data Store 预览要求

首先验证您的环境是否满足 Astra Data Store 预览要求。

Astra Data Store 预览版既支持裸机部署，也支持基于 VM 的部署。Astra Data Store 预览集群可以在具有四个或更多工作节点的 Kubernetes 集群上运行。Astra Data Store 预览软件可以与在同一 Kubernetes 集群上运行的其他应用程序共存。

Astra Data Store 预览版仅支持使用 Astra Trident CSI 驱动程序为 Kubernetes 工作负载配置永久性卷。未来版本的 Astra Data Store 将支持 VM 工作负载。

如果您计划从 Astra 控制中心管理 Astra Data Store 预览集群，请确保您的 Astra Data Store 预览集群符合 "由 Astra 控制中心管理的集群的要求" 除了此处概述的要求之外。

Kubernetes 工作节点资源要求

以下是在 Kubernetes 集群中每个工作节点上分配给 Astra Data Store 预览软件所需的资源要求：

资源	最小值	最大值
数据驱动器的数量	• 3 （存在单独的缓存设备） • 4 （如果不存在缓存设备）	14
数据驱动器大小	100 GiB	4 TiB
可选缓存设备的数量	1 （大小为 8 GiB 或更大）	不适用
vCPU 数量	10	10
RAM	35 GiB	35 GiB

为了获得最佳写入性能，您应配置一个专用的高持久性，低延迟，低容量缓存设备。

每个工作节点都有以下附加要求：

- 主机磁盘（启动）上要存储 Astra Data Store 预览日志文件的可用空间为 100 GiB 或更大。
- 至少为集群，数据和管理流量提供一个 10GbE 或更快的网络接口。此外，还可以使用一个额外的 1GbE 或更快的接口来隔离管理流量。

硬件和软件要求

已在以下硬件平台，软件和存储配置上验证 Astra Data Store 预览软件。请访问 "NetApp 社区支持" 如果您的 Kubernetes 集群配置不同。

硬件平台

- HPE DL360

- HPE DL380
- Dell R640
- Dell R740

Astra Data Store 预览版已通过以下驱动器类型的验证：

- * 裸机部署 *： Astra Data Store 预览版直接安装在没有任何虚拟机管理程序的 Linux 集群上的 Kubernetes 集群上
  - NVMe TLC SSD
- * 基于 VM 的部署 *：在 ESXi 集群上托管的 Linux VM 上的 Kubernetes 集群上安装 Astra Data Store 预览版
  - 基于 SAS 或 NVMe TLC SSD 的数据存储库
  - 驱动器显示为虚拟磁盘或直通驱动器

如果主机在硬件 RAID 控制器后使用 SSD ，请将硬件 RAID 控制器配置为使用 " 直通 " 模式。

每个驱动器都应具有唯一的序列号。在虚拟机创建期间，在虚拟机高级设置中添加属性 `disk.enableid=true` 。

软件

- 虚拟机管理程序： Astra Data Store 预览版已通过基于 VMware 的 VM 部署与 ESXi 7.0 的验证。Astra Data Store 预览版不支持基于 KVM 的部署。
- 已在以下主机操作系统上验证 Astra Data Store 预览版：
  - Red Hat Enterprise Linux 8.4
  - Red Hat Enterprise Linux 8.2
  - Red Hat Enterprise Linux 7.9
  - Red Hat Enterprise Linux CoreOS （ RHCOS ）
  - CentOS 8
  - Ubuntu 20.04
- Astra Data Store 预览版已通过以下 Kubernetes 分发版的验证：
  - Red Hat OpenShift 4.7
  - Google Anthos 1.7
  - Kubernetes 1.21
  - Kubernetes 1.20

Astra Data Store 预览版需要使用 Astra Trident 21.10.1 版进行存储配置和编排。请参见 "Astra Trident 安装说明"。

网络要求

Astra Data Store 预览版要求每个集群为 MVIP 配置一个 IP 地址。它必须是与 MIP 位于同一子网中的未使用或未配置的 IP 地址。Astra Data Store 预览管理界面应与 Kubernetes 节点的管理界面相同。

此外，还可以按下表所述配置每个节点：

此表使用了以下缩写词： mip： management IP address cip： cluster ip address MVIP： management virtual IP address

Configuration	所需的 IP 地址
每个节点一个网络接口	• 每个节点两（ 2 ）个： ◦ MIP/CIP ：每个节点的管理接口上有一（ 1 ）个预配置的 IP 地址 ◦ Data IP ：在与 MIP 相同的子网中，每个节点一（ 1 ）个未使用或未配置的 IP 地址
每个节点两个网络接口	• 每个节点三个： ◦ MIP ：每个节点的管理接口上有一（ 1 ）个预配置的 IP 地址 ◦ CIP ：在与 MIP 不同的子网中，每个节点的数据接口上有一（ 1 ）个预先配置的 IP 地址 ◦ Data IP ：在与 CIP 相同的子网中，每个节点一（ 1 ）个未使用或未配置的 IP 地址

对于这两种配置，应省略集群自定义资源（ CR ）文件 `asadscluster.YAML` 中的数据网络网关字段。每个节点上的现有路由配置可容纳所有地址。

这些配置中不使用任何 VLAN 标记。

Astra Trident

Astra Data Store 预览版要求应用程序 Kubernetes 集群运行 Astra Trident 21.10.1 。Astra Data Store 预览版可配置为 "存储后端" 使用 Astra Trident 配置永久性卷。

CNI 配置

Astra 数据存储预览已通过以下 CNI 的验证：

- 适用于 Vanilla Kubernetes 集群的 Calico 和 Weave Net CNI
- 适用于 Red Hat OpenShift 容器平台（ OCP ）的 OpenShift SDN
- Cilium for Google Anthos

这些 CNI 要求禁用主机防火墙（ firewalld ）。

永久性卷共享要求

每个 Astra Data Store 预览集群都支持使用永久性卷来满足该集群上安装的任何应用程序的存储需求。对于 Astra Data Store 预览版中的永久性卷，请考虑以下要求：

要求

- NFSv4.1 客户端 / 服务器必须安装在 Kubernetes 集群上。
- 必须在工作节点上安装 nfs-utils 软件包。
- Kubernetes 应用程序使用通过 NFSv4.1 共享的永久性卷访问文件，这需要使用 AUTH_SYS 身份验证方法。

许可

要获得完整功能， Astra Data Store 预览版需要获得 Astra Data Store 预览许可证。 "请在此处注册" 以获取 Astra Data Store 预览许可证。注册后，系统将向您发送许可证下载说明。

AutoSupport 配置

Astra 数据存储预览版要求启用 AutoSupport 并连接到 AutoSupport 后端。这可以通过直接 Internet 访问或代理配置来实现。

。 "用于发送强制遥测 AutoSupport 捆绑包的定期设置" 不应更改。如果禁用定期发送 AutoSupport 捆绑包，则集群将被锁定，并且无法创建新卷，直到再次启用定期设置为止。

下一步行动

查看 "快速入门" 概述。

快速启动 Astra Data Store 预览

此页面简要概述了开始使用 Astra Data Store 预览版所需的步骤。每个步骤中的链接将转到一个页面，其中提供了更多详细信息。

试用！如果要尝试使用 Astra Data Store 预览版，可以使用 90 天预览许可证。

"请在此处注册" 以获取 Astra Data Store 预览许可证。

跨度 **class="image><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png" Alt-one">** 查看 **Kubernetes** 集群要求

- 集群必须以运行状况良好的状态运行，并且至少具有四个或更多工作节点。
- 属于 Astra Data Store 预览部署的每个 Kubernetes 工作节点都应具有相同接口类型（ SATA ， SAS 或 NVMe ）的 SSD ，并具有相同数量的驱动器，这些驱动器将分配给 Astra Data Store 预览集群。
- 每个 SSD 都应具有唯一的序列号。

了解更多信息 "Astra Data Store 预览要求"。

跨度 **class="image><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png" Alt-twe">** 下载并安装 **Astra Data Store** 预览版

- 从下载 Astra Data Store 预览版 "NetApp 支持站点"。

- 在本地环境中安装 Astra Data Store 预览版。
- 应用 Astra Data Store 预览许可证。
- 安装 Astra Data Store 预览集群。
- 配置 Astra Data Store 预览监控。
- 如果您使用的是 Red Hat OpenShift ，请在 Red Hat OpenShift 容器平台（ OCP ）上安装 Astra Data Store preview 。

了解更多信息 "安装 Astra Data Store 预览版"。

跨度 **class="image><img src="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png" Alt-Three ** 完成一些初始设置任务

- 安装 Astra Trident 。
- 安装 Kubernetes Snapshot 自定义资源定义（ CRD ）和控制器。
- 将 Astra Data Store 预览设置为存储后端。
- 创建默认的 Astra Data Store 预览存储类。

详细了解 "初始设置过程"。

设置完 Astra Data Store 预览后，您接下来可能会执行以下操作：

- 使用 kubectl 命令和 kubectl astrad 扩展来管理集群，包括将节点置于维护模式，更换驱动器或更换节点等任务。了解更多信息 "如何在 Astra Data Store 预览版中使用 kubectl 命令"。
- 配置监控端点。了解更多信息 "配置监控端点"。

"安装 Astra Data Store 预览版"。

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"。

设置 Astra Data Store 预览组件

安装 Astra Data Store 预览版并满足某些环境前提条件后，您将安装 Astra Trident ，配置 Kubernetes 快照功能，设置存储后端并创建默认存储类：

- [Install Astra Trident]
- [Install Kubernetes snapshot CRDs and Controller]
- [Set up Astra Data Store as storage backend]
- [Create a default Astra Data Store storage class]

安装 Astra Trident

对于 Astra Data Store 预览版，您需要安装 Astra Trident 21.10.1 。您可以使用以下选项之一安装 Astra Trident ：

- "使用 tridentctl 安装 Astra Trident"。
- "使用 Trident 操作员安装 Astra Trident"。

> 您可以手动或使用 Helm 部署 Trident 操作员。

安装 Kubernetes Snapshot CRD 和控制器

要创建永久性卷声明（ PVC ）快照，需要使用 Kubernetes Snapshot CRC 和控制器。如果尚未为您的环境安装 CRD 和控制器，请运行以下命令进行安装。

> 以下命令示例假设 ` /trident` 作为目录；但是，您使用的目录可以是用于下载 YAML 文件的任何目录。

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"。
- 下载 "Kubernetes 快照控制器 YAML 文件"：
  - Setup-snapshot-controller.yaml
  - rbac 快照控制器 .yaml
- 下载 "YAML CRD"：
  - snapshot.storage.k8s.io_volumesnapshotclasses.yaml
  - snapshot.storage.k8s.io_volumesnapshotcontents.yaml
  - snapshot.storage.k8s.io_volumesnapshots.yaml

步骤

1. 应用 snapshot.storage.k8s.io_volumesnapshotclasses.yaml ：

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

响应：

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshotclasses.snap
shot.storage.k8s.io created
```

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

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

响应：

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

3. 应用 snapshot.storage.k8s.io_volumesnapshots.yaml：

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

响应：

```
customresourcedefinition.apiextensions.k8s.io/volumesnapshots.snapshot.s
torage.k8s.io created
```

4. 应用 setup-snapshot-controller.yaml：

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

响应：

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

5. 应用 rbac 快照控制器 .yaml：

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

响应：

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

6. 验证是否已应用 CRD YAML 文件：

```
kubectl get crd | grep volumesnapshot
```

响应示例：

```
astradsvolumesnapshots.astrads.netapp.io              2021-08-
04T17:48:21Z
volumesnapshotclasses.snapshot.storage.k8s.io         2021-08-
04T22:05:49Z
volumesnapshotcontents.snapshot.storage.k8s.io        2021-08-
04T22:05:59Z
volumesnapshots.snapshot.storage.k8s.io               2021-08-
04T22:06:17Z
```

7. 验证是否已应用快照控制器文件：

```
kubectl get pods -n kube-system | grep snapshot
```

响应示例：

```
snapshot-controller-7f58886ff4-cdh78
1/1     Running   0          13s
snapshot-controller-7f58886ff4-tmrd9
1/1     Running   0          32s
```

将 **Astra** 数据存储设置为存储后端

在 ads_backend.json 文件中配置存储后端参数，并创建 Astra Data Store 存储后端。

步骤

1. 使用安全终端创建 `ads_backend.json`：

```
vi ads_backend.json
```

2. 配置 JSON 文件：

   a. 将 ` "cluster"` 值更改为 Astra Data Store 集群的集群名称。

   b. 将 ` "namespace"` 值更改为要用于创建卷的命名空间。

   c. 将 ` "autoExportPolicy"` 值更改为 `true` ，除非为此后端设置了 exportpolicy CR 。

   d. 使用要授予访问权限的 IP 地址填充 ` "autosExportCIDRs"` 列表。使用 `0.0.0.0/0` 允许所有。

   e. 对于 ` "kubeconfig"` 值，请执行以下操作：

      i. 将 .Kube/config YAML 文件转换为不含空格的 JSON 格式并将其最小化：

转换示例：

```
python3 -c 'import sys, yaml, json;
json.dump(yaml.load(sys.stdin), sys.stdout, indent=None)' <
~/.kube/config > kubeconf.json
```

ii. 编码为 base64 ，并使用 base64 输出作为 ` "kubeconfig"` 值：

示例编码：

```
cat kubeconf.json | base64 | tr -d '\n'
```

```
{
    "version": 1,
    "storageDriverName": "astrads-nas",
    "storagePrefix": "",
    "cluster": "example-1234584",
    "namespace": "astrads-system",
    "autoExportPolicy": true,
    "autoExportCIDRs": ["0.0.0.0/0"],
    "kubeconfig": "<base64_output_of_kubeconf_json>",
    "debugTraceFlags": {"method": true, "api": true},
    "labels": {"cloud": "on-prem", "creator": "trident-dev"},
    "defaults": {
        "qosPolicy": "bronze"
    },
    "storage": [
        {
            "labels": {
                "performance": "extreme"
            },
            "defaults": {
                "qosPolicy": "bronze"
            }
        },
        {
            "labels": {
                "performance": "premium"
            },
            "defaults": {
                "qosPolicy": "bronze",
            }
        },
        {
            "labels": {
                "performance": "standard"
            },
            "defaults": {
                "qosPolicy": "bronze"
            }
        }
    ]
}
```

3. 更改为下载 Trident 安装程序的目录：

```
cd <trident-installer or path to folder containing tridentctl>
```

4. 创建存储后端：

```
./tridentctl create backend -f ads_backend.json -n trident
```

响应示例：

```
+------------------+----------------
+--------------------------------------+--------+---------+
|       NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------+----------------
+--------------------------------------+--------+---------+
| example-1234584  | astrads-nas    | 2125fa7a-730e-43c8-873b-
6012fcc3b527 | online |       0 |
+------------------+----------------
+--------------------------------------+--------+---------+
```

创建默认的 **Astra Data Store** 存储类

创建 Astra Trident 默认存储类并将其应用于存储后端。

步骤

1. 创建 trident CSI 存储类：

   a. 创建 ads_sc_example.yaml：

   ```
   vi ads_sc_example.yaml
   ```

   响应：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: true
mountOptions:
  - vers=4
```

b. 创建 trident CSI：

```
kubectl create -f ads_sc_example.yaml
```

响应：

```
storageclass.storage.k8s.io/trident-csi created
```

2. 验证是否已添加存储类：

```
kubectl get storageclass -A
```

响应：

```
NAME          PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
trident-csi   csi.trident.netapp.io   Delete          Immediate
true                   6h29m
```

3. 更改为下载 Trident 安装程序的目录：

```
cd <trident-installer or path to folder containing tridentctl>
```

4. 验证 Astra Trident 后端是否已使用默认存储类参数进行更新：

```
./tridentctl get backend -n trident -o yaml
```

响应示例：

```
items:
- backendUUID: 2125fa7a-730e-43c8-873b-6012fcc3b527
  config:
    autoExportCIDRs:
    - 0.0.0.0/0
    autoExportPolicy: true
    backendName: ""
    cluster: example-1234584
    credentials: null
    debug: false
    debugTraceFlags:
      api: true
      method: true
    defaults:
      exportPolicy: default
      qosPolicy: bronze
      size: 1G
      snapshotDir: "false"
      snapshotPolicy: none
    disableDelete: false
    kubeconfig: <ID>
    labels:
      cloud: on-prem
      creator: trident-dev
    limitVolumeSize: ""
    namespace: astrads-system
    nfsMountOptions: ""
    region: ""
    serialNumbers: null
    storage:
    - defaults:
        exportPolicy: ""
        qosPolicy: bronze
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
      region: ""
      supportedTopologies: null
      zone: ""
    - defaults:
        exportPolicy: ""
        qosPolicy: bronze
        size: ""
```

```
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: premium
      region: ""
      supportedTopologies: null
      zone: ""
    - defaults:
        exportPolicy: ""
        qosPolicy: bronze
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: standard
      region: ""
      supportedTopologies: null
      zone: ""
    storageDriverName: astrads-nas
    storagePrefix: ""
    supportedTopologies: null
    version: 1
    zone: ""
  configRef: ""
  name: example-1234584
  online: true
  protocol: file
  state: online
  storage:
    example-1234584_pool_0:
      name: example-1234584_pool_0
      storageAttributes:
        backendType:
          offer:
          - astrads-nas
        clones:
          offer: true
        encryption:
          offer: false
        labels:
          offer:
            cloud: on-prem
            creator: trident-dev
            performance: extreme
        snapshots:
          offer: true
```

```
      storageClasses:
      - trident-csi
      supportedTopologies: null
    example-1234584_pool_1:
      name: example-1234584_pool_1
      storageAttributes:
        backendType:
          offer:
          - astrads-nas
        clones:
          offer: true
        encryption:
          offer: false
        labels:
          offer:
            cloud: on-prem
            creator: trident-dev
            performance: premium
        snapshots:
          offer: true
      storageClasses:
      - trident-csi
      supportedTopologies: null
    example-1234584_pool_2:
      name: example-1234584_pool_2
      storageAttributes:
        backendType:
          offer:
          - astrads-nas
        clones:
          offer: true
        encryption:
          offer: false
        labels:
          offer:
            cloud: on-prem
            creator: trident-dev
            performance: standard
        snapshots:
          offer: true
      storageClasses:
      - trident-csi
      supportedTopologies: null
  volumes: []
```

Astra 数据存储预览限制

Astra 数据存储是 Kubernetes 原生的共享文件软件定义存储（ SDS ）解决方案，适用于内部数据中心，可帮助客户管理其云原生应用程序。

Astra Data Store 预览版具有以下资源限制。

资源	最小值	最大值
Astra Data Store 预览集群中的节点数	4.	5.
每个节点的永久性卷数	不适用	10
每个节点的永久性卷的总已配置容量	不适用	1 TiB
卷大小	20 MiB	1 TiB
每个卷的快照数	0	256
每个卷的克隆数	0	9

Astra Data Store 预览版不支持 VM 工作负载。VMware VVol 工作负载支持将在未来版本中提供。

Astra Data Store 预览会限制性能，不应用于性能特征化。

下一步行动

确保您的配置符合 "要求"。

有关 Astra Data Store 预览版的常见问题

查找有关 Astra Data Store 预览版的安装，配置，升级和故障排除的常见问题解答。

一般问题

- 是否可以使用 Astra Data Store 预览版进行生产？ * 否虽然 Astra Data Store 的设计和开发旨在提供企业级弹性，但作为预览版， Astra Data Store 预览版不适用于生产工作负载。
- 是否可以对虚拟机工作负载使用 Astra Data Store 预览版？ * Astra Data Store 预览版仅限于在 Kubernetes 上运行的应用程序，无论是裸机还是虚拟机。未来版本将支持 Kubernetes 上以及直接在 ESXi 虚拟机上运行的应用程序。请参见 "Astra 数据存储要求"。
- Astra Data Store 预览版的运行是否依赖于其他 NetApp 产品？ *

是的。Astra Data Store preview 要求在工作负载 Kubernetes 集群上部署 NetApp CSI 驱动程序 Astra Trident 21.10.1 及更高版本。了解相关信息 "Astra 数据存储要求"。

使用 Astra Data Store 预览集群作为存储后端的应用程序可以使用 "Astra 控制中心" 21.12 版可利用应用程序感知型数据管理功能，包括数据保护，灾难恢复和 Kubernetes 工作负载迁移。

- 如何管理 Astra Data Store 预览集群？ * 您可以使用 kubectl 命令和 Kubernetes API 扩展来管理 Astra Data Store 预览资产。

`kubectl astrad` 命令包含一个 ` -h` 交换机，可为您提供使用情况和标志文档。

- 如何监控 Astra 数据存储预览集群指标？ * 您可以使用 Cloud Insights 监控 Astra 数据存储预览指标。请参见 "使用 Cloud Insights 监控指标"。

您还可以监控日志。请参见 "配置和监控事件日志"。

- 我是否可以在 Kubernetes 集群中将 Astra 数据存储预览与 ONTAP 或其他存储提供程序结合使用？ * 是。Astra Data Store 预览版可与应用程序集群中的其他存储提供程序结合使用。
- 如果从 Astra Data Store 预览版中删除 Kubernetes 集群，则是否会卸载 Astra Trident ？ * 如果卸载 Astra Data Store 预览版，则不会从集群中卸载 Astra Trident 。如果需要卸载 Astra Trident ，则必须单独执行此操作。

许可

- Astra Data Store 预览版是否需要许可证？ * 是， Astra Data Store 预览版需要 NetApp 许可证文件（ NLF ）。

请参见 "Astra 数据存储要求"。

- Astra Data Store 预览许可证有效多长时间？ * Astra Data Store 预览许可证的默认期限为自下载日期起 90 天。

在 **Kubernetes** 集群上安装和使用 **Astra Data Store** 预览版

- 是否可以在裸机或虚拟机上运行的 Kubernetes 集群上安装 Astra Data Store 预览版？ * 是。Astra Data Store 预览版可以安装在裸机上运行的 Kubernetes 集群或 ESXi 虚拟机上。请参见 "Astra Data Store 预览要求"。
- 支持哪些版本的 Kubernetes for Astra Data Store 预览版？ *

Astra Data Store 预览版适用于与 v1.20 及更高版本兼容的 Kubernetes 分发版。但是，目前尚未对所有 Kubernetes 分发版进行验证。了解相关信息 "Astra Data Store 预览要求"。

- 我的 Kubernetes 集群超过 5 个工作节点。是否可以在其中安装 Astra Data Store 预览版？ * 是。Astra Data Store 预览集群可以部署在 Kubernetes 集群中的 4 个工作节点上。部署完成后，您可以将集群扩展到 5 个辅助节点。
- Astra Data Store 预览版是否支持从专用注册表脱机安装？ * 是。可以从本地注册表脱机安装 Astra Data Store 预览版。请参见 "安装 Astra Data Store 预览版"。但是， Astra 数据存储预览版需要连接（直接或通过代理）到 NetApp AutoSupport 后端（ support.netapp.com ）才能继续运行。
- 使用 Astra Data Store 预览版是否需要 Internet 连接？ * Astra Data Store 预览版需要连接到 NetApp AutoSupport 后端，以便定期发送强制的遥测 AutoSupport 捆绑包。此连接可以是直接连接，也可以通过代理进行连接。如果缺少此连接或禁用了 AutoSupport ，则集群将锁定，并且新卷创建将被禁用，直到恢复定期上传的捆绑包为止。
- Astra Data Store preview 使用了哪些角色和特权？ * 您需要成为 Kube 管理员才能部署 Astra Data Store 预览运算符。

Astra Data Store preview 具有一个名为 `astrads-ds-nodeinfo-asadsversion` 的特权取消命名集，用于

发现在选择节点时使用的节点资源。

此外，操作员还将使用特权 Kubernetes 作业在选定工作节点上安装存储集群的容器，以构建 Astra Data Store 预览存储集群。

- 在安装 Astra Data Store 预览版时需要更新哪些清单文件？ * 来自从下载的 Astra Data Store 预览版软件包 "NetApp 支持站点"，您将获得以下清单：
- astradscluster.yaml
- astradsoperator.yaml
- astradsversion.yaml
- monitoring_operator.yaml

您需要使用特定于部署的配置更新 `astradscluster.yaml` 清单。请参见 "安装 Astra Data Store 预览版"。

故障排除和支持

借助 Astra Data Store 预览版，您可以使用 NetApp 容器可宽延通道访问社区支持。此渠道由 NetApp 支持和我们的技术营销工程师监控。

"NetApp 容器 Slack 通道"

预览版要求您的系统连接到云并集成到 NetApp Active IQ 和 AutoSupport 工具中。

请参见 "Astra 数据存储支持操作"。

- 如何提出支持案例或要求对快速问题进行澄清？ * 要提出支持案例或获得有关快速问题的澄清，请在上报告您的问题描述或问题 "NetApp 容器 Slack 通道"。NetApp 支持部门将与您密切合作，尽最大努力提供帮助。
- 如何申请新功能？ * 如果您对支持的配置或功能有任何疑问，请联系 astra.feedback@netapp.com 。
- 如何生成支持日志包？ * 请参见 "生成支持包" 有关为 Astra Data Store 预览版设置和下载支持日志包的说明。
- Astra Data Store 预览无法找到我的 Kubernetes 节点。如何修复此问题？ * 请参见 "安装 Astra Data Store 预览版"。
- IPv6 地址是否可用于管理，数据和集群网络？ * 否， Astra Data Store 预览版仅支持 IPv4 地址。未来版本的 Astra Data Store 预览版将增加 IPv6 支持。
- 在 Astra Data Store 预览版上配置卷时使用的是什么 NFS 版本？ * 默认情况下，对于为 Kubernetes 应用程序配置的所有卷， Astra Data Store 预览版支持 NFS v4.1 。
- 为什么即使我已为 Astra Data Store 预览版配置了大容量驱动器，也无法获得更大的永久性卷？ * Astra Data Store 预览版将为 Astra Data 中所有节点上的所有卷配置的最大容量限制为 1 TiB ，最多 5 TiB 存储预览集群。

请参见 "Astra Data Store 预览要求"。

升级 **Astra Data Store** 预览版

- 是否可以从 Astra Data Store 预览版升级？ * 否Astra Data Store 预览版不适用于生产工作负载，新版本的 Astra Data Store 预览版软件需要全新安装。