入门

Astra Data Store

NetApp
June 02, 2022

目录

入门

Astra 数据存储要求

首先验证您的环境是否满足Astra Data Store要求。

Astra数据存储既支持裸机部署、也支持基于VM的部署。Astra数据存储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软件可以与在同一Kubernetes集群上运行的其他应用程序共存。

- [Kubernetes worker node resources]
- [Hardware and software]
- [Networking]
- Astra Trident
- [Container Network Interfaces]
- [Licensing]

如果您计划从Astra控制中心管理Astra数据存储集群、请确保您的Astra数据存储集群符合 "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

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

资源	最小值	最大值
Astra数据存储集群中的节点数	4.	16.
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 (8 GiB大小)	不适用

Astra数据存储支持每个Kubernetes工作节点的以下vCPU和RAM组合：

- 9个vCPU、RAM为38 GiB
- 23个vCPU、94 GiB RAM

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：

已在以下硬件平台、软件和存储配置上验证Astra Data Store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

- Dell R640
- Dell R740
- HPE DL360
- HPE DL380
- 联想SR630
- 联想SR650

基于VM的部署还可以使用上列为兼容的服务器 "VMware 兼容性指南"。

存储

Astra数据存储已通过SATA、SAS和NVMe TLC SSD的验证。

对于基于VM的部署、您可以使用以虚拟磁盘或直通驱动器形式提供的驱动器。

- 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已通过使用vSphere 7.0部署基于VMware的虚拟机的验证。Astra数据存储不支持基于KVM的部署。
- 已在以下主机操作系统上验证Astra数据存储：
  - 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数据存储已通过以下Kubernetes分发版的验证：
  - Red Hat OpenShift 4.6到4.9
  - Google Anthos 1.8到1.10
  - Kubernetes 1.20至1.23

- Rancher RKE1.3.3

Astra数据存储需要使用Astra Trident进行存储配置和编排。支持Astra Trident 21.10.1至22.04版。请参见 "Astra Trident 安装说明"。

网络

Astra数据存储要求每个集群为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数据存储管理接口应与Kubernetes节点的管理接口相同。

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

这些配置中不使用任何 VLAN 标记。

Astra Trident

Astra数据存储要求应用程序Kubernetes集群运行Astra Trident进行存储配置和编排。支持Astra Trident 21.10.1至22.04版。Astra数据存储可配置为 "存储后端" 使用 Astra Trident 配置永久性卷。

容器网络接口

Astra数据存储已通过以下容器网络接口(CNI)的验证。

- 适用于RKE集群的Calico
- 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

> - 随Cilium CNI部署的Astra数据存储需要portmap插件才能支持hostPort。您可以通过将`cnis-canding-mode：portmap`添加到cilial-config configMap并重新启动Cilium Pod来启用CNI链模式。
> - 只有Calico和OpenShift SDN CNI才支持启用了防火墙的配置。

许可

要启用全部功能、Astra数据存储需要有效的许可证。

"请在此处注册" 以获取Astra Data Store许可证。注册后，系统将向您发送许可证下载说明。

下一步行动

查看 "快速入门" 概述。

有关详细信息 ...

"Astra数据存储限制"

Astra 数据存储快速入门

此页面简要概述了开始使用Astra数据存储所需的步骤。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！如果要试用Astra数据存储、可以使用90天评估许可证。 "请在此处注册" 获取Astra Data Store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集群。
- 每个 SSD 都应具有唯一的序列号。

了解更多信息 "Astra 数据存储要求"。

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**数据存储

- 从下载Astra数据存储 "NetApp 支持站点"。
- 在本地环境中安装Astra数据存储。
- 应用Astra Data Store许可证。
- 安装Astra数据存储集群。

了解更多信息 "安装Astra数据存储"。

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数据存储设置为存储后端。
- 创建默认的Astra Data Store存储类。
- 为遥测服务配置Astra数据存储。

详细了解 "初始设置过程"。

完成Astra数据存储的设置后、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 "如何对Astra数据存储使用kubectl命令"。
- 配置监控端点。了解更多信息 "配置监控端点"。
- "将Astra数据存储与VMware结合使用"。

"安装 Astra 数据存储"。

安装 Astra 数据存储

您可以使用Kubernetes本机命令或使用Astra控制中心中的UI安装Astra数据存储。

安装选项

- 使用**Kubernetes**本机命令：要使用Kubernetes本机命令安装Astra Data Store、请完成中所述的安装步骤 此操作步骤。
- 使用**Astra**控制中心：要使用Astra控制中心安装Astra数据存储库、请完成中所述的安装步骤 此操作步骤。

您将需要什么

- "开始安装之前、请为部署Astra Data Store准备您的环境"。
- 访问 "NetApp 支持站点"。 "注册" 如果您还没有完全访问的NetApp支持站点帐户、请下载评估版。
- 答 "NetApp 许可证文件（ NLF ）" 用于Astra数据存储。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数据存储使用。

安装 **Astra** 数据存储

标准Kubernetes集群的Astra数据存储安装过程将指导您完成以下高级步骤。此外、还介绍了适用于Red Hat OpenShift容器平台(OCP)环境的其他安装步骤。

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

要将Astra Data Store与Google Anthos结合使用、请完成以下安装步骤、然后将Astra Data Store集群添加到您的Anthos环境中。

要在Rancher RKE环境中安装Astra数据存储、请完成以下安装步骤、并将Rancher命令替换为kubectl命令。

下载**Astra Data Store**软件包并提取映像

1. 登录到 "NetApp 支持站点" 并下载Astra数据存储捆绑包(`Astra_Data_Store_2022.05.tar`)。

   如果要查找早期版本的捆绑包的说明、请参见 "该版本的文档"。

2. （可选）使用以下命令验证捆绑包的签名：

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

3. 创建目录：

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

4. 提取映像：

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

   这些图像将提取到工作目录中创建的`astrads/images`目录中。

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(在以下示例中使用`/usr/bin/`作为路径)。kubectl-astrad是一个自定义kubectl扩展、用于安装和管理Astra数据存储集群。

使用`which kubectl`命令查找安装kubectl二进制文件的路径。

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

2. 将Astra Data Store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数据存储映像的注册表路径；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 $(ls astrads/images/*.tar) ; do
   astraImage=$(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}'~'
./astrads/manifests/*.yaml
```

OpenShift 操作步骤

以下操作步骤 仅用于在Red Hat OpenShift容器平台(OCP)上部署。跳过此操作步骤 以在非OCP Kubernetes集群上部署。

- 创建命名空间以部署Astra数据存储
- 创建自定义 SCC
- 创建角色和角色绑定

示例 **1.** 详细信息

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

只有在Red Hat OpenShift容器平台(OCP)上部署时、才需要执行以下步骤。

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
```

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集群Pod所需的权限。使用此操作步骤 为Astra数据存储提供所需的权限(在示例中列出)。

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

只有在 Red Hat OpenShift 容器平台（ OCP ）上部署时，才需要执行以下步骤。

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Astra数据存储的默认服务帐户要使用的所需角色和角色绑定。

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

只有在 Red Hat OpenShift 容器平台（ OCP ）上部署时，才需要执行以下步骤。

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
```

配置私有映像注册表

对于某些环境、您可以将配置更改为从使用机密的专用注册表中提取映像。

1. 创建`astrads-system`命名空间、除非您已在上一步中创建了此命名空间：

```
kubectl create namespace astrads-system
```

2. 创建密钥：

```
kubectl create secret docker-registry <secret-name> -n astrads-system
--docker-server=<registry name> --docker-username= <registry username>
--docker-password=<registry user password>
```

3. 向服务帐户添加机密配置信息：

```
kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name":
"<secret-name>"}]}' -n astrads-system
```

这些更改将在您使用时应用 安装Astra数据存储操作员。

安装**Astra**数据存储操作员

1. 列出Astra数据存储清单：

```
ls astrads/manifests/*yaml
```

响应：

```
astrads/manifests/monitoring_operator.yaml
astrads/manifests/astradscluster.yaml
astrads/manifests/astradsversion.yaml
astrads/manifests/astradsoperator.yaml
astrads/manifests/vasa_asup_certs.yaml
astrads/manifests/manifest.yaml
astrads/manifests/configuration.yaml
```

2. 使用`kubectl apply`部署操作员：

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

响应：

根据您执行的是标准安装还是、命名空间响应可能会有所不同 "OpenShift容器平台安装"。

```
namespace/astrads-system created
customresourcedefinition.apiextensions.k8s.io/astradsadddrives.astrads.n
etapp.io 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/astradsexportpolicies.astr
ads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrad
s.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradskeyproviders.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/astradsnodemanagements.ast
rads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsqospolicies.astrads
.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradssearkeyrotatereques
ts.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsversions.astrads.ne
tapp.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-astrads-system-admin-role created
role.rbac.authorization.k8s.io/astrads-astrads-system-reader-role
created
role.rbac.authorization.k8s.io/astrads-astrads-system-writer-role
created
role.rbac.authorization.k8s.io/astrads-leader-election-role created
role.rbac.authorization.k8s.io/astrads-manager-role created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-admin-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-reader-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astrads-writer-clusterrole
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsautosupport-viewer-
```

```
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-astradsexportpolicy-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsexportpolicy-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsfaileddrive-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-astradsnfsoption-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnfsoption-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodeinfo-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsnodemanagement-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsqospolicy-viewer-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-editor-role
created
clusterrole.rbac.authorization.k8s.io/astrads-astradsversion-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-astradsvolumefile-editor-
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumefile-viewer-
```

```
role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
editor-role created
clusterrole.rbac.authorization.k8s.io/astrads-astradsvolumesnapshot-
viewer-role created
clusterrole.rbac.authorization.k8s.io/astrads-manager-role created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-rolebinding
created
rolebinding.rbac.authorization.k8s.io/astrads-leader-election-
rolebinding created
rolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-admin-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-reader-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-astrads-writer-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/astrads-manager-rolebinding
created
configmap/astrads-autosupport-cm created
configmap/astrads-firetap-cm created
configmap/astrads-kevents-asup created
configmap/astrads-metrics-cm created
secret/astrads-autosupport-certs created
secret/astrads-webhook-server-cert created
service/astrads-webhook-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版本YAML

1. 使用`kubectl apply`进行部署：

```
kubectl apply -f ./astrads/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-2jqnk                     1/1     Running   0
2m7s
astrads-ds-nodeinfo-dbk7v                     1/1     Running   0
2m7s
astrads-ds-nodeinfo-rn9tt                     1/1     Running   0
2m7s
astrads-ds-nodeinfo-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许可证

1. 应用从NetApp获得的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
e100000006-ads-capacity astrads-example-cluster  true  Astra Data Store
true        2023-01-23  2022-04-04T14:38:54Z
```

安装**Astra**数据存储集群

1. 打开 YAML 文件：

```
vim ./astrads/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：

      - 根据您的许可证和Astra Data Store安装大小、将`adsNodeConfig`值更改为您的安装所需的值：
        - 小型：9个CPU和38个内存
        - 中：23个CPU和94个内存
      - (可选)删除围绕`adsNodeSelector`部分的注释。如果要将Astra数据存储限制为仅安装在选定的工作节点池上、请配置此设置。
      - (可选)指定在4-16之间应在Astra数据存储集群中使用的特定节点数。
      - 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数据存储节点的数量相同。对于Astra数据存储、这意味着至少需要4个地址、如果您计划稍后扩展集群、则最多需要16个地址。
      - 在 `adsDataNetworks` 中，指定数据网络使用的网络掩码。
      - 在 `adsNetworkInterfaces` 中，将 ` <mgmt_interface_name>` 和 ` <cluster_and_storage_interface_name>` 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。此外、请务必删除`adsNetworkInterfaces`部分的注释。

        集群和存储网络必须位于同一接口上。Astra数据存储管理接口应与Kubernetes节点的管理接口相同。

   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 。

为了简明起见、我们从以下YAML示例中删除了一些注释。

```
apiVersion: astrads.netapp.io/v1beta1
kind: AstraDSCluster
metadata:
  name: astrads-cluster-name
  namespace: astrads-system
spec:
  adsNodeConfig:
    cpu: 9
    memory: 38
  # [Optional] Specify node selector labels to select the nodes for
creating ADS cluster
  # adsNodeSelector:
  #   matchLabels:
  #     customLabelKey: customLabelValue
  adsNodeCount: 4
  mvip: ""
  adsDataNetworks:
    - addresses: ""
      netmask:
  # Specify the network interface names to use for management, cluster
and storage networks.
  # If none are specified, the node's primary interface will be used for
management, cluster and storage networking.
  # To move the cluster and storage networks to a different interface
than management, specify all three interfaces to use here.
  # NOTE: The cluster and storage networks need to be on the same
interface.
  adsNetworkInterfaces:
    managementInterface: "<mgmt_interface_name>"
    clusterInterface: "<cluster_and_storage_interface_name>"
    storageInterface: "<cluster_and_storage_interface_name>"
  # [Optional] Provide a 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 ./astrads/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-gcmj8                           1/1       Running   1
12d
astrads-ds-nodeinfo-j826x                           1/1       Running   3
12d
astrads-ds-nodeinfo-vdthh                           1/1       Running   3
12d
astrads-ds-nodeinfo-xwgsf                           1/1       Running   0
12d
astrads-ds-support-828vw                            2/2       Running   2
5d2h
astrads-ds-support-astrads-example-cluster-cfzts    2/2       Running   0
8h
astrads-ds-support-astrads-example-cluster-nzkkr    2/2       Running   15
7h49m
astrads-ds-support-astrads-example-cluster-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   2022.05.0-X  e100000006
10.x.x.x   13m
```

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	消息和重要性
已选择ControlPlaneNodesS	已成功选择【编号】控制平面节点以加入ADS集群。Astra数据存储操作员确定了具有CPU、内存、存储和网络的足够多节点、以创建Astra数据存储集群。
ADSClusterCreateInProProgress	Astra Data Store集群控制器已启动集群创建操作。
ADSClusterCreateSuccessess	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>
```

使用Astra控制中心安装Astra数据存储

要在Astra控制中心中部署和使用Astra数据存储、请执行以下操作。

您需要什么？ **#8217**；将需要什么

- 您已查看 Astra数据存储的一般前提条件。
- 您已安装Astra控制中心。

步骤

1. "使用Astra控制中心部署Astra数据存储"。

下一步行动

- * Kubernetes本机部署和第三方分发版*：通过执行其他操作完成Astra Data Store部署 "设置任务"。
- * Astra控制中心*：如果您已使用Astra控制中心部署Astra数据存储、则无需遵循这些原则 "设置任务" 除非您要配置任何其他监控选项。部署Astra数据存储后、您可以使用Astra控制中心UI完成以下任务：
  - "监控Astra数据存储资产的运行状况"。
  - "管理Astra Data Store后端存储"。
  - "监控节点，磁盘和永久性卷声明（ PVC ）"。

设置Astra数据存储组件

你先请 "安装了独立的Astra数据存储库" 并解决了一些问题 "环境前提条件"、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：

如果您使用Astra控制中心部署Astra数据存储、则无需执行这些步骤、除非您打算进行设置 其他监控选项。

- [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]
- [Configure Astra Data Store monitoring]

安装 Astra Trident

对于Astra数据存储、您需要安装Astra Trident 21.10.1或更高版本。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、请为部署Astra Data Store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 configured
```

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 configured
```

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 configured
```

4. 应用 setup-snapshot-controller.yaml：

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

响应：

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

5. 应用 rbac 快照控制器 .yaml：

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

响应：

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

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：

   以下步骤将提供一个JSON示例。

   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": "silver"
    },
    "storage": [
        {
            "labels": {
                "performance": "extreme"
            },
            "defaults": {
                "qosPolicy": "gold"
            }
        },
        {
            "labels": {
                "performance": "premium"
            },
            "defaults": {
                "qosPolicy": "silver"
            }
        },
        {
            "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
   ```

   示例

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  creationTimestamp: "2022-05-09T18:05:21Z"
  name: ads-silver
  resourceVersion: "3361772"
  uid: lo023456-da4b-51e3-b430-3aa1e3bg111a
mountOptions:
- vers=4
parameters:
  backendType: astrads-nas
  selector: performance=premium
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

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
```

响应：

```
NAME          PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
ads-silver    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: gold
        size: ""
        snapshotDir: ""
        snapshotPolicy: ""
      labels:
        performance: extreme
      region: ""
```

```
      supportedTopologies: null
      zone: ""
    - defaults:
        exportPolicy: ""
        qosPolicy: silver
        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:
      - ads-silver
      supportedTopologies: null
  volumes: []
```

配置Astra数据存储监控

(可选)您可以配置Astra数据存储以供其他遥测服务监控。如果您不使用Astra控制中心进行Astra数据存储监控、或者希望将监控扩展到其他端点、则建议使用此操作步骤。

如果您的Astra数据存储实例是独立部署、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。

对于Astra控制中心部署、监控操作员会自动进行配置。您可以跳过以下操作步骤 的前两个命令。

在设置监控之前、您需要在`astrads-system`命名空间中有一个活动的Astra数据存储集群。

步骤

1. 运行此安装命令：

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

2. 配置用于监控的Astra数据存储：

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

3. 配置Astra数据存储以将EMS日志流式传输到弹性端点：

```
kubectl astrads monitoring es --port <portname> --host <hostname>
```

Astra数据存储早期访问计划(EAP)版本限制

在早期访问计划期间、Astra数据存储具有以下资源限制。

资源	最小值	最大值
Astra数据存储集群中的节点数	4.	16.
每个节点的永久性卷数	不适用	50
卷大小	20 MiB	2 TiB
每个卷的快照数	0	1023
每个卷的克隆数	0	9

资源	最小值	最大值
每个节点的VM数	0	50

有关Astra数据存储的常见问题

查找有关Astra Data Store Early Access计划版本的安装、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数据存储早期访问计划版本进行生产？*否虽然Astra数据存储库的设计和开发旨在提供企业级故障恢复能力、但Astra数据存储库的早期访问计划版本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数据存储？*是。Astra数据存储同时支持Kubernetes和VMware VVol工作负载。

请参见 "了解采用VMware的Astra Data Store"。

*是否可以使用VMware vSphere管理Astra数据存储？*是的、可以使用适用于VMware vSphere的NetApp Astra插件在vCenter中本机管理Astra数据存储。请参见 "管理VMware安装的Astra Data Store组件"。

- Astra数据存储是否依赖于其他NetApp产品才能正常运行？*

是的。Astra Data Store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"。

要启用VMware工作流和适用于VMware vSphere的NetApp Astra插件、需要使用Astra控制中心。

使用Astra Data Store集群作为存储后端的应用程序可以使用 "Astra 控制中心" 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数据存储集群？*您可以使用kubectl命令和Kubernetes API扩展来管理Astra数据存储资源。

`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数据存储集群指标？*您可以使用Cloud Insights 监控Astra数据存储指标(请参见 "使用 Cloud Insights 监控指标")或Kubernetes原生 监控(请参见 "使用Prometheus和Grafana进行监控"）。

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数据存储与ONTAP 或其他存储提供程序结合使用？*是。Astra数据存储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数据存储库中删除Kubernetes集群、则是否会卸载Astra Trident？*如果卸载Astra数据存储库、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数据存储是否需要许可证？*是的、Astra数据存储需要早期访问计划的评估版NetApp许可证文件(NLF)。

请参见 "Astra 数据存储要求"。

- Astra Data Store评估许可证有效期有多长时间？* Astra Data Store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数据存储

*是否可以在运行于裸机或虚拟机上的Kubernetes集群上安装Astra数据存储？*是。Astra数据存储可以安装在裸机上运行的Kubernetes集群或vSphere VM上。请参见 "Astra 数据存储要求"。

支持哪些版本的**Kubernetes for Astra Data Store**?

Astra数据存储可与v1.20及更高版本兼容的Kubernetes分发版配合使用。但是，目前尚未对所有 Kubernetes 分发版进行验证。了解相关信息 "Astra 数据存储要求"。

*我的Kubernetes集群超过4个工作节点。是否可以在其中安装Astra数据存储？*是。Astra数据存储集群最初必须部署在Kubernetes集群中的4个辅助节点上。部署完成后、您可以将集群最多扩展到16个辅助节点。

- Astra数据存储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数据存储。请参见 "安装 Astra 数据存储"。

*是否需要Internet连接才能使用Astra Data Store？*否、Astra Data Store Early Access Program不需要Internet连接。但是、建议连接到NetApp AutoSupport 后端、以便定期发送遥测捆绑包。此连接可以是直接连接，也可以通过代理进行连接。

- Astra数据存储使用哪些角色和特权？*您需要成为Kube管理员才能部署Astra数据存储操作员。

Astra数据存储具有一个名为`astrads-ds-nodeinfo`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、操作员还将使用具有权限的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存储集群。

从下载的Astra Data Store捆绑包中*安装Astra Data Store需要更新哪些清单文件？* "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 数据存储"。

故障排除和支持

借助Astra Data Store、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"

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数据存储设置和下载支持日志包的说明。
- Astra数据存储无法找到我的Kubernetes节点。如何修复此问题？ * 请参见 "安装 Astra 数据存储"。
- IPv6地址是否可用于管理、数据和集群网络？*否、Astra数据存储仅支持IPv4地址。未来版本的Astra Data Store将增加对IPv6的支持。

*在Astra Data Store上配置卷时使用的是什么NFS版本？*对于为Kubernetes应用程序配置的所有卷、Astra Data Store支持NFS v4.1、而对于为VMware工作负载配置的所有卷、则支持NFSv3。

请参见 "Astra 数据存储要求" 和 "Astra数据存储限制"。

升级**Astra**数据存储

*是否可以从Astra Data Store预览版升级？*是。您可以从Astra数据存储早期访问计划版本升级到未来版本。