



# **Astra Data Store 预览文档**

## **Astra Data Store**

NetApp  
February 09, 2022

This PDF was generated from <https://docs.netapp.com/zh-cn/astra-data-store/index.html> on February 09, 2022. Always check docs.netapp.com for the latest.

# 目录

Astra Data Store 预览文档 .....	1
发行说明 .....	2
此版本的 Astra 数据存储中的新增功能 .....	2
已知问题 .....	3
已知限制 .....	3
概念 .....	4
Astra 数据存储预览简介 .....	4
Astra Data Store 预览部署模式 .....	6
集群扩展 .....	7
Astra Data Store 预览版中的存储效率 .....	8
Astra Data Store 中的安全性预览 .....	8
入门 .....	10
Astra Data Store 预览要求 .....	10
快速启动 Astra Data Store 预览 .....	13
Astra 数据存储安装概述 .....	14
设置 Astra Data Store 预览组件 .....	39
Astra 数据存储预览限制 .....	50
有关 Astra Data Store 预览版的常见问题 .....	50
使用 Astra 数据存储 .....	53
使用 kubectl 命令管理 Astra Data Store 预览资产 .....	53
部署测试应用程序 .....	59
管理集群 .....	63
使用 Cloud Insights 监控指标 .....	75
配置和监控事件日志 .....	87
将 Astra 控制中心与 Astra Data Store 预览结合使用 .....	88
使用自动脚本卸载 Astra Data Store 预览版 .....	89
卸载不带脚本的 Astra Data Store 预览版 .....	95
知识和支持 .....	101
故障排除 .....	101
获取帮助 .....	101
自动支持监控 .....	101
法律声明 .....	106

# Astra Data Store 预览文档

# 发行说明

我们很高兴地宣布，Astra Data Store 预览版为 21.12 预览版。

- ["此版本的 Astra Data Store 预览版中的新增功能"](#)
- ["已知问题"](#)
- ["已知限制"](#)

在 Twitter @NetAppDoc 上关注我们。通过成为发送有关文档的反馈 ["GitHub 贡献者"](#) 或发送电子邮件至 [doccomments@netapp.com](mailto:doccomments@netapp.com)。

## 此版本的 **Astra** 数据存储中的新增功能

我们很高兴地宣布，Astra Data Store 预览版提供了 2021.12.1 修补程序。

### 2022 年 2 月 8 日（2021.12.1）

适用于 Astra Data Store 预览版（21.12）的修补版本（2021.12.1）。

- 在此版本中，Astra Data Store preview 支持使用 Calico CNI 的 VXLAN 配置。
- Astra Data Store 预览版现在支持启用了 Calico WireGuard 的网络配置。
- 更新的 AstraDSCluster.YAML adsNetworkInterfaces 一节提供了经过改进的描述性注释。
- 现在，我们提供了新的 Astra Data Store 卸载脚本，作为 kubectl 命令卸载过程的一种更简单的替代方法。

### 2021 年 12 月 21 日（21.12）

初始版本的 Astra Data Store 预览版。

- ["它是什么"](#)
- ["部署模式和组件"](#)
- ["入门所需的资源"](#)
- ["安装"](#) 和
- ["管理"](#) 和 性能
- ["使用 Cloud Insights 进行监控"](#)
- ["获取帮助"](#) 和

## 了解更多信息

- ["已知问题"](#)
- ["已知限制"](#)
- ["NetApp 知识库文章"](#)

# 已知问题

已知问题用于确定可能会阻止您成功使用此产品预览版的问题。

如果 **Pod** 处于崩溃环路中，使用默认活跃度探针值的 **MongoDB** 部署将失败作为临时决策，请在 MongoDB 部署规范中将活跃度探针 `initialDelaySeconds` 设置为 600 秒。

## Astra 数据存储预览卸载过程可能会使发生原因 **Pod** 保持终止状态

Kubernetes v1.20 中的 Astra 数据存储预览卸载过程有时可能会使发生原因 Pod 保持终止状态。

请参见 ["手动卸载不使用脚本的 Astra Data Store 预览版"](#) 或。

## 了解更多信息

- ["已知限制"](#)

# 已知限制

已知限制表示此产品预览版不支持或无法与之正确交互的平台，设备或功能。仔细审查这些限制。

## 不支持删除一个或多个节点的功能

Astra Data Store 预览版支持更换发生故障的节点，但不支持节点删除功能。

## 不支持添加或删除驱动器的功能

Astra Data Store 预览版支持更换故障驱动器，但不支持在现有集群中添加或删除驱动器。

## 启用防火墙后，未验证 **Astra Data Store** 预览功能

Astra Data Store 预览版要求禁用主机 `firewalld` 。尚未验证使用 `Calico HostEndpoint` 等 CNI 工具启用防火墙。

## 升级或修补程序需要全新安装

Astra Data Store 预览版不适用于生产工作负载。

## 基于 **Ubuntu** 的裸机或虚拟机直通部署需要 **NVMe TLC SSD**

此限制不适用于基于 RHEL ， RHCOS 或 CentOS 的部署。

## 了解更多信息

- ["已知问题"](#)

# 概念

## Astra 数据存储预览简介

Astra 数据存储预览版是 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 数据存储为容器和 VM 工作负载提供原生共享文件服务，并提供 NetApp 企业数据管理。

使用 Astra Data Store 预览版，您可以执行以下操作：

- \* 支持 Kubernetes 容器化工作负载 \*：使用您习惯使用的企业级数据管理服务和工具。
- \* 将 Kubernetes 的 "应用程序即服务" 平台用于开发运营 \*：创建弹性的软件定义自助式平台，以提供自动化，可重复的服务，消除 developers 带来的复杂性。

## Astra 产品系列

Astra 系列产品可提供 Kubernetes 应用程序数据生命周期管理，从而简化有状态应用程序的操作。轻松保护，备份和迁移 Kubernetes 工作负载，并即时创建有效的应用程序克隆。

Astra 产品包括：

- \* Astra Control\*：使用应用程序感知型数据管理工具来管理，保护和移动公有云和内部环境中的 Kubernetes 工作负载。
  - \* Astra 控制服务\*：使用 NetApp 管理的服务对公有云中的 Kubernetes 工作负载进行数据管理。
  - \* Astra 控制中心\*：使用自管理软件对内部 Kubernetes 工作负载进行数据管理。
- \* Astra Data Store preview\*：对容器和 VM 工作负载使用 Kubernetes 本机共享文件服务进行企业数据管理。
- \* Astra Trident\*：与 NetApp 存储提供商合作，为 Kubernetes 工作负载配置和管理符合容器存储接口（CSI）要求的存储。

请参见 ["Astra 系列简介"](#)。



## Astra Data Store 预览功能

Astra Data Store preview 可通过以下功能为您的云原生 applications 提供端到端 Kubernetes 本机存储和数据管理：

- \* Kubernetes 本机共享文件服务 \*：使用标准 NFS 客户端作为容器和 VM 的统一数据存储库，为 Kubernetes 提供共享文件服务原生。
- \* 云扩展 \*：在同一资源池中提供 Kubernetes 本机多个并行文件系统，以实现类似于云的扩展和利用率，从而无需在集群之外单独管理存储。
- \* API 优先方法 \*：将基础架构作为代码交付用于自动化集群和工作负载 management。
- \* 企业级数据管理 \*：提供自动化应用程序感知型数据保护和灾难恢复：

- \* NetApp 技术 \*：利用 NetApp 数据管理技术执行快照，备份，复制和克隆，以便用户可以在 Kubernetes 中构建和部署企业级应用程序。
- \* 故障恢复能力 \*：对 Kubernetes 本机工作负载使用复制和擦除编码技术，以提高故障恢复能力。
- \* 数据效率 \*：通过实时重复数据删除和数据压缩功能进行扩展时控制成本。
- \* 适合您的现有环境 \*：支持基于微服务的工作负载和传统工作负载，为主要 Kubernetes 分发版提供服务，提供文件存储，并在您选择的 hardware 上运行。
- \* 与 NetApp Cloud Insights \* 集成：为持续 optimization 提供可观察性，分析和监控功能。

## 开始使用 **Astra Data Store** 预览版

首先，"[了解 Astra 数据存储的要求](#)"。

然后，"[开始使用](#)"。

### 有关详细信息 ...

- "[Astra 系列简介](#)"
- "[Astra Control Service 文档](#)"
- "[Astra 控制中心文档](#)"
- "[Astra Trident 文档](#)"
- "[使用 Astra Control API](#)"
- "[Cloud Insights 文档](#)"
- "[ONTAP 文档](#)"

## **Astra Data Store** 预览部署模式

Astra Data Store 预览版使用与 Kubernetes 部署和协调的应用程序直接管理主机上的存储驱动器。

您可以使用以下选项之一在裸机或虚拟服务器上安装 Astra Data Store 预览版：

- 部署在一个独立的专用 Kubernetes 集群上，为在单独集群（独立集群）中运行的 Kubernetes 应用程序提供永久性卷。
- 在 Kubernetes 集群上部署，同时在一节点池（融合集群）上托管其他工作负载应用程序。
- 在 Kubernetes 集群上部署，同时在其他节点池（离散集群）上托管其他工作负载应用程序。

["了解有关 Astra Data Store 硬件要求的更多信息"](#)。

Astra Data Store 预览是 Astra 产品系列的一部分。要了解有关整个 Astra 系列的观点，请参见 "[Astra 系列简介](#)"。

## **Astra Data Store** 预览生态系统

Astra Data Store 预览版可用于以下内容：



- **\* Astra 控制中心 \***：使用 Astra 控制中心软件对内部环境中的 Kubernetes 集群进行应用程序感知型数据管理。轻松备份 Kubernetes 应用程序，将数据迁移到其他集群以及即时创建有效的应用程序克隆。

Astra 控制中心支持具有 ONTAP 的 Astra Trident 存储后端或 Astra 数据存储预览存储后端的 OpenShift Kubernetes 集群。

- **\* Astra Trident \***：作为由 NetApp 维护的完全受支持的开源存储配置程序和编排程序，Astra Trident 使您能够为 Docker 和 Kubernetes 管理的容器化应用程序创建存储卷。

您可以使用 Astra Trident 在 Astra Data Store 预览版上创建卷。

- **\* Cloud Insights \***：Cloud Insights 是一款 NetApp 云基础架构监控工具，可用于监控由 Control 管理的 Kubernetes 集群的性能和利用率。Cloud Insights 将存储使用量与工作负载相关联。

在 Astra Control 中启用 Cloud Insights 连接后，遥测信息将显示在 Astra Control UI 页面中。Cloud Insights 显示有关在 Astra 数据存储预览版中管理的资源的信息。

## Astra Data Store 预览界面

您可以使用不同的界面完成任务：

- **\* Web 用户界面（UI） \***：Astra 控制服务和 Astra 控制中心使用同一个基于 Web 的 UI，您可以在其中管理，迁移和保护应用程序。此 UI 还会显示有关 Astra Data Store 预览卷的信息。
- **\* API \***：Astra 控制服务和 Astra 控制中心使用相同的 Astra 控制 API。使用 API，您可以执行与使用 UI 相同的任务。您还可以使用 Astra Control API 检索有关 Astra Data Store 预览的信息。
- **\* kubectl 命令 \***：要使用 Astra Data Store 预览版，您可以直接使用 kubectl 命令。
- **\* Kubernetes 扩展 \***：此外，您还可以使用 Astra Data Store 预览 Kubernetes API 扩展。

自定义资源定义（CRD）是 Kubernetes REST API 的扩展，该 API 是在部署 Astra Data Store 预览运算符时创建的。外部实体通过调用 Kubernetes API 服务器与 CRD 进行交互。Astra Data Store 预览版会监控特定 CRD 的更新，然后调用内部 REST API。

## 有关详细信息 ...

- ["Astra 系列简介"](#)
- ["Astra Control Service 文档"](#)
- ["Astra 控制中心文档"](#)
- ["Astra Trident 文档"](#)
- ["使用 Astra Control API"](#)
- ["Cloud Insights 文档"](#)
- ["ONTAP 文档"](#)

## 集群扩展

Astra Data Store 预览版支持集群中不同类型和功能的节点。如果要扩展集群，则 Astra Data Store 预览版支持添加具有任何性能功能的节点，只要这些节点不低于集群中性能最低的节点即可。新节点的存储容量必须与现有

节点相同。所有节点（包括扩展期间的新节点）至少需要满足中的最低要求 ["Astra Data Store 预览要求"](#)。

## 有关详细信息 ...

- ["Astra 系列简介"](#)
- ["Astra Control Service 文档"](#)
- ["Astra 控制中心文档"](#)
- ["Astra Trident 文档"](#)
- ["使用 Astra API"](#)
- ["Cloud Insights 文档"](#)
- ["ONTAP 文档"](#)

## Astra Data Store 预览版中的存储效率

Astra 数据存储预览版使用基于 NetApp ONTAP 和 SolidFire 技术的存储效率技术，包括：

- **\* 精简配置 \***：精简配置卷是指不预先预留存储的卷。而是根据需要动态分配存储。删除卷或 LUN 中的数据后，可用空间将释放回存储系统
- **\* 零块检测和消除 \***：采用精简配置的 ONTAP 存储系统可以检测卷中已置零的区域，以便回收该空间并将其用于其他位置。
- **\* 数据压缩 \***：数据压缩通过将数据块组合到数据压缩组中来减少卷所需的物理存储量，每个数据块都存储为一个块。与传统压缩方法相比，压缩数据的读取速度更快，因为 ONTAP 仅解压缩包含所请求数据的压缩组，而不是解压缩整个文件。
- **\* 重复数据删除 \***：重复数据删除可通过丢弃重复块并将其替换为对单个共享块的引用来减少卷（或 AFF 聚合中的所有卷）所需的存储量。读取经过重复数据删除的数据通常不会对性能产生任何影响。除了过载的节点之外，写入所产生的费用可以忽略不计。

默认情况下，所有这些功能均处于启用状态。

请参见 ["存储效率详细信息"](#)。

## 有关详细信息 ...

- ["Astra 系列简介"](#)
- ["Astra Control Service 文档"](#)
- ["Astra 控制中心文档"](#)
- ["Astra Trident 文档"](#)
- ["使用 Astra Control API"](#)
- ["ONTAP 文档"](#)

## Astra Data Store 中的安全性预览

Astra Data Store 预览版可通过多种方法确保客户端和管理员对存储的访问安全，保护通信和数据以及防止病

毒。

Astra Data Store 预览版使用以下安全方法：

- 使用相互传输层安全（MTLS）进行通信加密
- 基于角色的访问控制，用于控制对功能的访问
- 部署安全性
- 证书管理
- 空闲软件加密，包括内部密钥管理

有关详细信息 ...

- ["Astra 系列简介"](#)
- ["Astra Control Service 文档"](#)
- ["Astra 控制中心文档"](#)
- ["Astra Trident 文档"](#)
- ["使用 Astra Control API"](#)
- ["Cloud Insights 文档"](#)
- ["ONTAP 文档"](#)

## 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	<ul style="list-style-type: none"><li>• 3 （存在单独的缓存设备）</li><li>• 4 （如果不存在缓存设备）</li></ul>	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	<ul style="list-style-type: none"><li>• 每个节点两（2）个：<ul style="list-style-type: none"><li>◦ MIP/CIP：每个节点的管理接口上有一（1）个预配置的 IP 地址</li><li>◦ 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</li></ul></li></ul>
每个节点两个网络接口	<ul style="list-style-type: none"><li>• 每个节点三个：<ul style="list-style-type: none"><li>◦ MIP：每个节点的管理接口上有一（1）个预配置的 IP 地址</li><li>◦ 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</li><li>◦ 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</li></ul></li></ul>



对于这两种配置，应省略集群自定义资源（CR）文件 `asadscluster.YAML` 中的数据网络网关键字段。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。

## 有关详细信息 ...

["Astra 数据存储预览限制"](#)

## 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">&lt;img src="<a href="https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png" class="bare">https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-1.png"</a> Alt-one">&lt;/span>&lt; 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">&lt;img src="<a

[!\[\]\(082f818d99f166a3ba574d9284d73064\_img.jpg\)](https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-2.png) Alt-**Two** &lt;/span> 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](#)"。

[!\[\]\(3d8c13c92b853674f749aac6fa869926\_img.jpg\)](https://raw.githubusercontent.com/NetAppDocs/common/main/media/number-3.png) Alt-**Three** &lt;/span> 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](#)"。



## Astra Data Store deployment



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.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscLOUDsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslICENSES.astrads.netapp.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.netapp.io created
```

```
app.io created
customresourcedefinition.apiextensions.k8s.io/astradsvolumesnapshots.as-
trads.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_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。



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 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 k8s label key that defines which protection
domain a node belongs to.
# adsProtectionDomainKey: ""
# [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-lab0dbc-j9jzc	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-k9wp8	1/1	Running	0	5d1h
astrads-ds-astrads-cluster-lab0dbc-pwk42	1/1	Running	0	5d2h
astrads-ds-astrads-cluster-lab0dbc-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
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：

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

- [\[Configure Astra Data Store preview monitoring\]](#)
- [\[Install the monitoring operator\]](#)

### 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.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradscloudsnapshots.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsclusters.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsdeployments.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsexportpolicies.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradsfaileddrives.astrads.netapp.io created
customresourcedefinition.apiextensions.k8s.io/astradslicenses.astrads.netapp.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_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。



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 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 k8s label key that defines which protection
domain a node belongs to.
# adsProtectionDomainKey: ""
# [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-lab0dbc-j9jzc 1/1 Running 0 5d2h  
astrads-ds-astrads-cluster-lab0dbc-k9wp8 1/1 Running 0 5d1h  
astrads-ds-astrads-cluster-lab0dbc-pwk42 1/1 Running 0 5d2h  
astrads-ds-astrads-cluster-lab0dbc-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-nzkkv 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
```

### 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 CRD 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.snapshot.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.snapshot.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.storage.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：

```
astradvolumesnapshots.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 > kubeconfig.json
```

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

示例编码：

```
cat kubeconfig.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=4.1

```

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。

## 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](mailto: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](#)" 和 。

## 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。

# 使用 Astra 数据存储

## 使用 kubectl 命令管理 Astra Data Store 预览资产

您可以使用 kubectl 命令和 Kubernetes API 扩展来管理 Astra Data Store 预览资产。

要了解如何部署示例应用程序，请参见 ["部署测试应用程序"](#)。

有关以下集群维护信息，请参见 ["管理集群"](#)：

- 将节点置于维护模式
- 更换驱动器
- 添加节点
- 更换节点

您需要什么？ [#8217](#) ；将需要什么

- 您安装在中的 Astra Data Store 预览 kubectl 插件 ["安装 Astra Data Store 预览版"](#)

## 列出适用于 Astra Data Store 预览版的 Kubernetes 自定义 API 资源

您可以在 Kubernetes 中使用 kubectl 命令与 Astra Data Store 预览集群进行交互并观察其状态。

从 `api-resources` 命令中列出的每个项目都代表一个 Kubernetes 自定义资源定义（CRD），Astra Data Store 预览版可在内部使用该定义定义（CRD）来管理集群。

此列表对于获取每个 Astra Data Store 预览对象的短名称以减少键入效果特别有用，如后面所示。

1. 显示用于 Astra Data Store 预览的 Kubernetes 自定义 API 资源列表：

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

响应：

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
astradsversions	adsve	astrads.netapp.io	true	
AstraDSVersion				
astradsclusters	adscl	astrads.netapp.io	true	
AstraDSCluster				
astradslicenses	adsli	astrads.netapp.io	true	
AstraDSLICENSE				
astradsnodeinfoes	adsni	astrads.netapp.io	true	
AstraDSNodeInfo				
astradsvolumes	adsvo	astrads.netapp.io	true	
AstraDSVolume				
astradsqospolicies	adsqp	astrads.netapp.io	true	
AstraDSQosPolicy				
astradsexportpolicies	adsep	astrads.netapp.io	true	
AstraDSEExportPolicy				
astradsvolumesnapshots	adsvs	astrads.netapp.io	true	
AstraDSVolumeSnapshot				
astradsvolumefiles	adsvf	astrads.netapp.io	true	
AstraDSVolumeFiles				
astradsautosupports	adsas	astrads.netapp.io	true	
AstraDSAutoSupport				
astradsfaileddrives	adsfd	astrads.netapp.io	true	
AstraDSFailedDrive				
astradsnodemanagements	adsnm	astrads.netapp.io	true	
AstraDSNodeManagement				

2. 要获取 Kubernetes 集群中所有当前的 Astra Data Store 预览对象，请使用 `kubectl get ADS -a` 命令：

```
kubectl get ads -A
```

响应：

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

NAMESPACE	NAME	STATUS	VERSION	SERIAL NUMBER	MVIP	AGE
astrads-system	astradscluster.astrads.netapp.io/	created	arda-9.11.1	e000000009	10.224.8.146	46h

NAMESPACE	NAME
-----------	------



```

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

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

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

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

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

NAMESPACE          NAME                                     ADSCLUSTER
VALID PRODUCT          EVALUATION ENDDATE    VALIDATED
astrads-system    astradslicense.astrads.netapp.io/e0    astrads-cluster-
9f1 true  Astra Data Store true          2022-02-07 2021-12-16T20:43:23Z

```

### 3. 使用以下短名称之一显示集群中卷的当前状态：

```
kubectl get adsvo -A
```

响应：

NAMESPACE	NAME	SIZE	IP	CLUSTER
astrads-system	test234	21Gi	172.25.138.109	astrads-cluster-9f1c99f true
astrads-system	test2345	21Gi	172.25.138.111	astrads-cluster-9f1c99f true

## 使用 **kubectl** 扩展上的 **help** 选项

`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。

### 1. 显示有关 Astra Data Store preview `kubectl` 扩展中所有命令的帮助：

```
kubectl astrads -h
```

响应：

```
A kubectl plugin for inspecting your AstraDS deployment

Usage:
  astrads [command]

Available Commands:
  asup          Manage AutoSupport
  clusters      Manage clusters
  drives        Manage drives in a cluster
  faileddrive   Manage drive replacement in a cluster
  help          Help about any command
  license       Manage license in the astrads cluster
  maintenance   Manage maintenance status of a node
  monitoring    Manage Monitoring Output
  nodes         Manage nodes in a cluster

Flags:
  --as string                Username to impersonate for the
operation
  --as-group stringArray     Group to impersonate for the
```

operation, this flag can be	repeated to specify multiple
groups.	
--cache-dir string	Default HTTP cache directory
	(default "/u/arda/.kube/http-
cache")	
--certificate-authority string	Path to a cert file for the
certificate authority	
--client-certificate string	Path to a client certificate file
for TLS	
--client-key string	Path to a client key file for TLS
--cluster string	The name of the kubeconfig
cluster to use	
--context string	The name of the kubeconfig
context to use	
-h, --help	help for astrads
--insecure-skip-tls-verify	If true, the server's
certificate will not be checked	
	for validity. This will make
your HTTPS connections insecure	
--kubeconfig string	Path to the kubeconfig file to
use for CLI requests.	
-n, --namespace string	If present, the namespace scope
for this CLI request	
--request-timeout string	The length of time to wait
before giving up on a single	
	server request. Non-zero values
should contain a	
	corresponding time unit (e.g.
1s, 2m, 3h).	
	A value of zero means don't
	(default "0")
-s, --server string	The address and port of the
Kubernetes API server	
--token string	Bearer token for authentication
to the API server	
--user string	The name of the kubeconfig user
to use	

2. 有关命令的详细信息, 请使用 `astrad [command] -help`。

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

响应:

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

Usage:

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

Examples:

```
# Control plane collection
```

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

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

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

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

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

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

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

NOTE:

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

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

Flags:

<pre>-c, --component string</pre>	Specify the component to collect: [storage , controlplane , vasaprovider, all]
<pre>-d, --duration int</pre>	Duration is the duration in hours from the startTime for collection
<pre>-e, --event string</pre>	of AutoSupport. This should be a positive integer
<pre>(default "manual")</pre>	Specify the callhome event to trigger.
<pre>-f, --forceUpload</pre>	Configure an AutoSupport to upload if
<pre>it is in the compressed state</pre>	and not
	uploading because it was created with
<pre>the 'local' option or if</pre>	automatic uploads of AutoSupports is
<pre>disabled</pre>	at the cluster level.
<pre>-h, --help</pre>	help for collect
<pre>-l, --local</pre>	Only collect and compress the
<pre>autosupport bundle. Do not upload</pre>	to support.
	Use 'download' to copy the collected

```

bundle after it is in
--nodes string          the 'compressed' state
                        Specify nodes to collect for storage
component. (default "all")
-t, --startTime string  StartTime is the starting time for
collection of AutoSupport.
                        This should be in the ISO 8601 date
time format.
                        Example format accepted:
                        2021-01-01T15:20:25Z, 2021-01-
01T15:20:25-05:00
-u, --usermessage string  UserMessage is the additional message
to include in the
                        AutoSupport subject.
                        (default "Manual event trigger from
CLI")

```

## 部署测试应用程序

以下是部署可与 Astra Data Store 预览版结合使用的测试应用程序的步骤。

在此示例中，我们使用 Helm 存储库从 BitNami 部署 MongoDB 图表。

您需要什么？ **#8217** ；将需要什么

- 部署和配置了 Astra Data Store 预览集群
- Trident 安装已完成

步骤

1. 从 BitNami 添加 Helm repo :

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

2. 部署 MongoDB :

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

3. 检查 MongoDB POD 的状态:

```

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

```

#### 4. 验证 MongoDB 使用的永久性卷声明（PVC）：

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

#### 5. 使用 kubectl 命令 get astradsvolume 列出卷：

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

#### 6. 使用 kubectl 命令 describe astradsvolume 描述卷：

```
~% kubectl describe astradsvolume pvc-1133453a-e2f5-48a5 -n astrads-
system
Name:          pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Namespace:     astrads-system
Labels:        astrads.netapp.io/cluster=jai-ads
               astrads.netapp.io/mip=10.192.1.39
               astrads.netapp.io/volumeUUID=cf33fd38-a451-596c-b656-
61b8270d2b5e
               trident.netapp.io/cloud=on-prem
               trident.netapp.io/creator=trident-dev
               trident.netapp.io/performance=premium
Annotations:   provisioning: {"provisioning":{"cloud":"on-
prem","creator":"trident-dev","performance":"premium"}}
               trident:
                 {"trident":{"version":"21.10.0-test.jenkins-trident-
stable-v21.10-
2+e03219ce37294d9ba54ec476bbe788c1a7772548","backendUUID":"","platform":
...
API Version:   astrads.netapp.io/v1alpha1
Kind:          AstraDSVolume
Metadata:
  Creation Timestamp:  2021-12-08T19:35:26Z
  Finalizers:
    trident.netapp.io/astradsvolume-finalizer
    astrads.netapp.io/astradsvolume-finalizer
  Generation:        1
  Managed Fields:
    API Version:      astrads.netapp.io/v1alpha1
```

```

Fields Type:  FieldsV1
fieldsV1:
  f:metadata:
    f:labels:
      f:astrads.netapp.io/cluster:
      f:astrads.netapp.io/mip:
      f:astrads.netapp.io/volumeUUID:
    f:status:
      .:
      f:cluster:
      f:conditions:
      f:created:
      f:displayName:
      f:exportAddress:
      f:internalName:
      f:mip:
      f:permissions:
      f:qosPolicy:
      f:requestedSize:
      f:restoreCacheSize:
      f:size:
      f:snapshotReservePercent:
      f:state:
      f:volumePath:
      f:volumeUUID:
Manager:      cluster-controller
Operation:    Update
Time:         2021-12-08T19:35:32Z
API Version:  astrads.netapp.io/v1alpha1
Fields Type:  FieldsV1
fieldsV1:
  f:status:
    f:exportPolicy:
Manager:      dms-controller
Operation:    Update
Subresource:  status
Time:         2021-12-08T19:35:32Z
API Version:  astrads.netapp.io/v1alpha1
Fields Type:  FieldsV1
fieldsV1:
  f:metadata:
    f:annotations:
      .:
      f:provisioning:
      f:trident:
    f:finalizers:

```

```

    v:"trident.netapp.io/astradsvolume-finalizer":
    f:labels:
      .:
      f:trident.netapp.io/cloud:
      f:trident.netapp.io/creator:
      f:trident.netapp.io/performance:
    f:spec:
      .:
      f:cluster:
      f:displayName:
      f:exportPolicy:
      f:noSnapDir:
      f:permissions:
      f:qosPolicy:
      f:size:
      f:snapshotReservePercent:
      f:type:
      f:volumePath:
    Manager:          trident_orchestrator
    Operation:        Update
    Time:             2021-12-08T19:35:34Z
    Resource Version: 12007115
    UID:              d522ae4f-e793-49ed-bbe0-9112d7f9167b
Spec:
  Cluster:           jai-ads
  Display Name:      pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  Export Policy:     pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
  No Snap Dir:       true
  Permissions:       0777
  Qos Policy:        silver
  Size:              9042036412
  Snapshot Reserve Percent: 5
  Type:              ReadWrite
  Volume Path:       /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Status:
  Cluster:  jai-ads
  Conditions:
    Last Transition Time: 2021-12-08T19:35:32Z
    Message:             Volume is online
    Reason:              VolumeOnline
    Status:              True
    Type:                AstraDSVolumeOnline
    Last Transition Time: 2021-12-08T19:35:32Z
    Message:             Volume creation request was successful
    Reason:              VolumeCreated
    Status:              True

```



```

Type: AstraDSVolumeCreated
Created: true
Display Name: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Export Address: 10.192.2.192
Export Policy: pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Internal Name: pvc_1133453a_e2f5_48a5_a06c_d14b8aa7be07
Mip: 10.192.1.192
Permissions: 777
Qos Policy: silver
Requested Size: 9042036412
Restore Cache Size: 0
Size: 8830116Ki
Snapshot Reserve Percent: 5
State: online
Volume Path: /pvc-1133453a-e2f5-48a5-a06c-d14b8aa7be07
Volume UUID: cf33fd38-a451-596c-b656-61b8270d2b5e
Events:
  Type      Reason           Age   From                      Message
  ----      -
  Normal    VolumeCreated    3m9s  ADSClusterController     Volume creation
request was successful

```

## 管理集群

您可以使用带有 Astra Data Store 预览版的 `kubectl` 命令来管理集群。

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

您需要什么？ **#8217** ；将需要什么

- 安装了 `kubectl` 和 `kubectl-astrad` 插件的系统。请参见 ["安装 Astra Data Store 预览版"](#)。

## 添加节点

要添加的节点应属于 Kubernetes 集群，并且其配置应与集群中的其他节点类似。

步骤

1. 如果新节点的 `dataIP` 尚未加入 `ADSCluster` CR，请执行以下操作：
  - a. 编辑 `astradscluster` CR 并在 `ADS Data Networks Addresses` 字段中添加额外的 `dataIP`：

```
~% kubectl edit astradscluster <cluster-name> -n astrads-system
```

响应:

```
adsDataNetworks:
  -addresses: dataIP1,dataIP2,dataIP3,dataIP4,*newdataIP*
```

- a. 保存 CR。
- b. 将节点添加到 Astra Data Store 预览集群:

```
~% kubectl astrads nodes add -cluster <cluster-name>
```

2. 否则, 只需添加节点:

```
~% kubectl astrads nodes add -cluster <cluster-name>
```

3. 验证是否已添加此节点:

```
~% kubectl astrads nodes list
```

## 将节点置于维护模式

需要执行主机维护或软件包升级时, 应将节点置于维护模式。



此节点必须已属于 Astra Data Store 预览集群。

当节点处于维护模式时, 您无法向集群添加节点。在此示例中, 我们会将节点 `nhcitj1525` 置于维护模式。

### 步骤

1. 显示节点详细信息:

```
~% kubectl get nodes
```

响应:

NAME	STATUS	ROLES	AGE	VERSION
nhcitjj1525	Ready	<none>	3d18h	v1.20.0
nhcitjj1526	Ready	<none>	3d18h	v1.20.0
nhcitjj1527	Ready	<none>	3d18h	v1.20.0
nhcitjj1528	Ready	<none>	3d18h	v1.20.0
scs000039783-1	Ready	control-plane,master	3d18h	v1.20.0

## 2. 确保节点尚未处于维护模式：

```
~% kubectl astrads maintenance list
```

响应（没有节点处于维护模式）：

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

## 3. 启用维护模式。

```
~% kubectl astrads maintenance create <cr-name> --node-name=<<node-name>> --variant=Node
```

示例：

```
~% kubectl astrads maintenance create maint1 --node-name="nhcitjj1525"
--variant=Node
Maintenance mode astrads-system/maint1 created
```

## 4. 列出节点。

```
~% kubectl astrads nodes list
```

响应：

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

## 5. 检查维护模式的状态：

```
~% kubectl astrads maintenance list
```

响应：

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

维护`模式下的 将以 `false 开头, 并更改为 true。M状态 从 PreparingForMaintenance 更改为 ReadyforMaintenance。

#### 6. 完成节点维护后, 禁用维护模式:

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

#### 7. 确保节点不再处于维护模式:

```
~% kubectl astrads maintenance list
```

## 更换节点

使用 kubectl 命令和 Astra Data Store 预览版替换集群中的故障节点。

### 步骤

#### 1. 列出所有节点:

```
~% kubectl astrads nodes list
```

响应:

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

#### 2. 描述集群:

```
~% kubectl astrads clusters list
```

响应:

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

### 3. 验证故障节点上的 Node HA 是否标记为 false：

```
~% kubectl describe astradscluster -n astrads-system
```

响应：

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Annotations:   kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"astrads.netapp.io/v1alpha1","kind":"AstraDSCluster","meta
data":{"annotations":{},"name":"cluster-multinodes-21209","namespa...
API Version:   astrads.netapp.io/v1alpha1
Kind:          AstraDSCluster

State:         Disabled
Variant:       None
Node HA:       false
Node ID:       4
Node Is Reachable: false
Node Management IP: 172.21.192.192
Node Name:     sti-rx2540-532d.ctl.gdl.englab.netapp.com
Node Role:     Storage
Node UUID:     6f6b88f3-8411-56e5-b1f0-a8e8d0c946db
Node Version:  12.75.0.6167444
Status:        Added
```

### 4. 通过将 'AddsNode Count' 的值减至 3，修改 astradscluster CR 以删除故障节点：

```
cat manifests/astradscluster.yaml
```

响应：

```
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
```

```

namespace: astrads-system
spec:
  # ADS Node Configuration per node settings
  adsNodeConfig:
    # Specify CPU limit for ADS components
    # Supported value: 9
    cpu: 9
    # Specify Memory Limit in GiB for ADS Components.
    # Your kubernetes worker nodes need to have at least this much RAM
    free
    # for ADS to function correctly
    # Supported value: 34
    memory: 34
    # [Optional] Specify raw storage consumption limit. The operator
    will only select drives for a node up to this limit
    capacity: 600
    # [Optional] Set a cache device if you do not want auto detection
    e.g. /dev/sdb
    # cacheDevice: ""
    # Set this regex filter to select drives for ADS cluster
    # drivesFilter: ".*"

    # [Optional] Specify node selector labels to select the nodes for
    creating ADS cluster
    # adsNodeSelector:
    #   matchLabels:
    #     customLabelKey: customLabelValue

    # Specify the number of nodes that should be used for creating ADS
    cluster
    adsNodeCount: 3

    # Specify the IP address of a floating management IP routable from any
    worker node in the cluster
    mvip: "172..."

    # Comma separated list of floating IP addresses routable from any host
    where you intend to mount a NetApp Volume
    # at least one per node must be specified
    # addresses: 10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4,10.0.0.5
    # netmask: 255.255.255.0
    adsDataNetworks:
      - addresses: "172..."
        netmask: 255.255.252.0

```

```

# [Optional] Provide a k8s label key that defines which protection
domain a node belongs to
# adsProtectionDomainKey: ""

# [Optional] Provide a monitoring config to be used to setup/configure
a monitoring agent.
monitoringConfig:
  namespace: "netapp-monitoring"
  repo: "docker.repo.eng.netapp.com/global/astra"

autoSupportConfig:
  # AutoUpload defines the flag to enable or disable AutoSupport
  upload in the cluster (true/false)
  autoUpload: true
  # Enabled defines the flag to enable or disable automatic
  AutoSupport collection.
  # When set to false, periodic and event driven AutoSupport
  collection would be disabled.
  # It is still possible to trigger an AutoSupport manually while
  AutoSupport is disabled
  # enabled: true
  # CoredumpUpload defines the flag to enable or disable the upload of
  coredumps for this ADS Cluster
  # coredumpUpload: false
  # HistoryRetentionCount defines the number of local (not uploaded)
  AutoSupport Custom Resources to retain in the cluster before deletion
  historyRetentionCount: 25
  # DestinationURL defines the endpoint to transfer the AutoSupport
  bundle collection
  destinationURL: "https://testbed.netapp.com/put/AsupPut"
  # ProxyURL defines the URL of the proxy with port to be used for
  AutoSupport bundle transfer
  # proxyURL:
  # Periodic defines the config for periodic/scheduled AutoSupport
  objects
  periodic:
    # Schedule defines the Kubernetes Cronjob schedule
    - schedule: "0 0 * * *"
    # PeriodicConfig defines the fields needed to create the
    Periodic AutoSupports
    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
```

##### 5. 验证是否已从集群中删除此节点:

```
~% kubectl get nodes --show-labels
```

响应:

NAME	STATUS	ROLES	AGE	VERSION
LABELS				
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
sti-rx2540-532d	Ready	<none>	24h	v1.20.0
sti-rx2540-533d	Ready	<none>	24h	

```
~% kubectl astrads nodes list
```

响应:

NODE NAME	NODE STATUS	CLUSTER NAME
sti-rx2540-534d	Added	cluster-multinodes-21209
sti-rx2540-535d	Added	cluster-multinodes-21209
sti-rx2540-536d	Added	cluster-multinodes-21209

```
~% kubectl get nodes --show-labels
```

响应:

NAME	STATUS	ROLES	AGE	VERSION
LABELS				
sti-astramaster-237	Ready	control-plane,master	24h	v1.20.0
beta.kubernetes.io/arch=amd64,				
sti-rx2540-532d	Ready	<none>	24h	v1.20.0
astrads.netapp.io/node-removal				

```
~% kubectl describe astradscluster -n astrads-system
```



响应:

```
Name:          cluster-multinodes-21209
Namespace:     astrads-system
Labels:        <none>
Kind:          AstraDSCluster
Metadata:
...
```

6. 通过修改集群 CR 将节点添加到集群以进行更换。节点数将递增至 4。验证是否已选取新节点进行添加。

```
rvi manifests/astradscluster.yaml
cat manifests/astradscluster.yaml
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSCluster
metadata:
  name: cluster-multinodes-21209
  namespace: astrads-system
```

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

响应:

```
astradscluster.astrads.netapp.io/cluster-multinodes-21209 configured
```

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

响应:

NAME	READY	STATUS	RESTARTS	AGE
astrads-cluster-controller...	1/1	Running	1	24h
astrads-deployment-support...	3/3	Running	0	24h
astrads-ds-cluster-multinodes-21209	1/1	Running		

```
~% kubectl astrads nodes list
```

响应:

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

```
~% kubectl astrads clusters list
```

响应:

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

```
~% kubectl astrads drives list
```

响应:

DRIVE NAME	DRIVE ID	DRIVE STATUS	NODE NAME	CLUSTER NAME
scsi-36000..	c3e197f2...	Active	sti-rx2540...	cluster-
multinodes-21209				

## 更换驱动器

当集群中的驱动器发生故障时，必须尽快更换驱动器以确保数据完整性。驱动器发生故障时，您将在集群 CR 节点状态，集群运行状况信息和指标端点中看到故障驱动器信息。

在 **nodeStatuss.driveStatuses** 中显示故障驱动器的集群示例

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

响应:

```

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

```

故障驱动器 CR 会在集群中自动创建，其名称与故障驱动器的 UUID 相对应。

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

响应：

```

...
apiVersion: astrads.netapp.io/v1alpha1
kind: AstraDSFailedDrive
metadata:
  name: c290a-5000-4652c-9b494
  namespace: astrads-system
spec:
  executeReplace: false
  replaceWith: ""
status:
  cluster: arda-6e4b4af
  failedDriveInfo:
    failureReason: AdminFailed
    inUse: false
    name: scsi-36000c290ace209465271ed6b8589b494
    path: /dev/disk/by-id/scsi-36000c290ace209465271ed6b8589b494
    present: true
    serial: 6000c290ace209465271ed6b8589b494
    node: sti-rx2540-300b.ctl.gdl.englab.netapp.com
  state: ReadyToReplace

```

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

响应:

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

步骤

1. 使用 `kubectl astrad show-replacements` 命令列出可能的替代驱动器, 该命令可筛选符合更换限制 (未在集群中使用, 未挂载, 无分区以及等于或大于故障驱动器) 的驱动器。

要在不筛选可能的替代驱动器的情况下列出所有驱动器, 请在 `show-replacements` 命令中添加 `-all`。

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

响应:

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

2. 使用 `replace` 命令将驱动器替换为已传递的序列号。如果 `-wait` 时间已过, 则命令将完成替换或失败。

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



如果使用不适当的 `-replaceWith` 序列号执行 `kubectl astrad faileddrive replace`, 则会显示类似以下内容的错误:

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

3. 要重新运行驱动器更换，请使用 `-force` 和上一个命令：

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

有关详细信息 ...

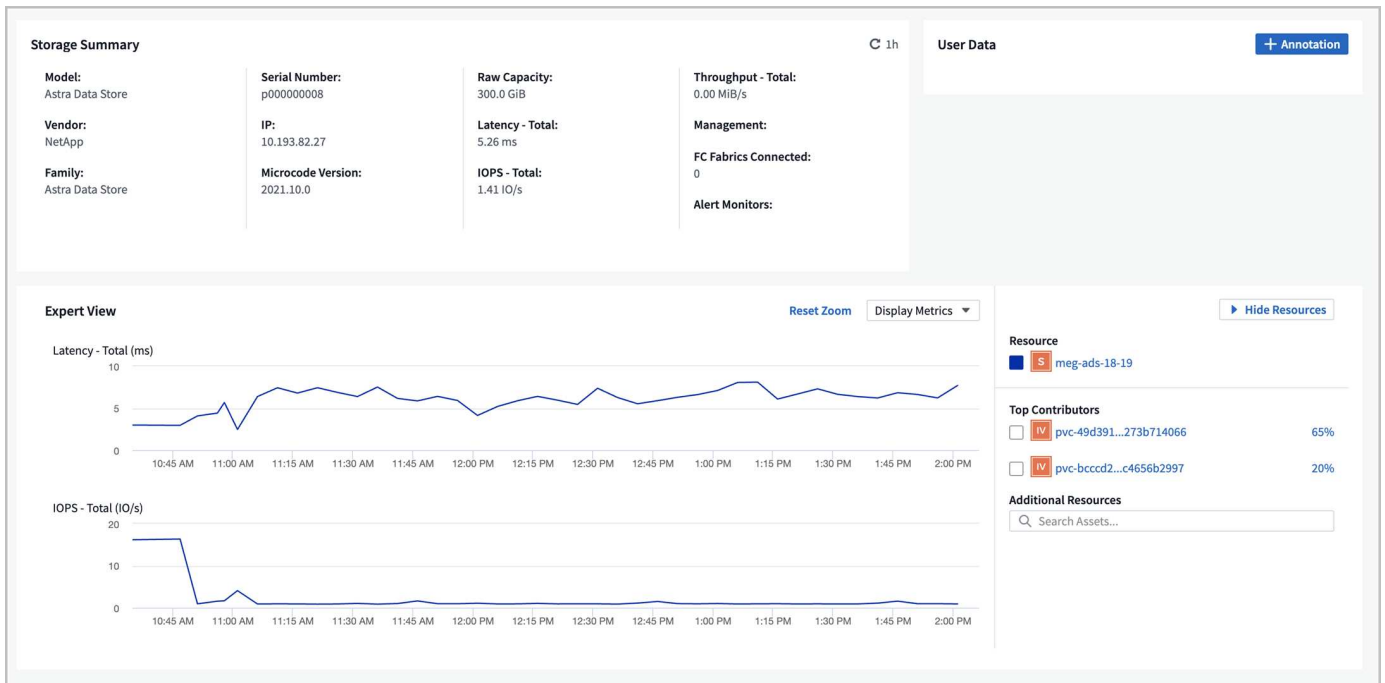
- ["使用 kubectl 命令管理 Astra Data Store 预览资产"](#)

## 使用 Cloud Insights 监控指标

您可以使用 Cloud Insights 监控 Astra 数据存储预览指标。

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

以下是 Cloud Insights 中显示的一些示例 Astra 数据存储预览指标：



您还可以使用显示在 Astra Data Store 预览版中生成的指标列表 [\[Open Metrics API help\]](#)。

## 完成 Cloud Insights 连接前提条件任务

在将 Astra 数据存储与 Cloud Insights 连接之前，您需要完成以下任务：

- "安装 Astra 数据存储监控操作员" 这是 Astra Data Store 预览安装说明的一部分。
- "安装 kubectl-astrad 二进制文件" 这是 Astra Data Store 预览安装说明的一部分。
- "创建 Cloud Insights 帐户"。
- 确保以下命令可用： `jk` ， `curl` ， `grep` 和 `JQ`

收集以下信息：

- 对以下类别具有读 / 写权限的 \* Cloud Insights API 访问令牌 \*：采集单元，数据收集，数据载入和日志载入。此选项将用于读 / 写操作，设置采集单元以及设置数据载入过程。
- \* Kubernetes API 服务器 IP 地址和端口 \*。用于监控 Astra Data Store 预览集群。
- \* Kubernetes API 令牌 \*。此选项用于调用 Kubernetes API。
- \* 永久性卷配置 \*。有关如何配置永久性卷的信息。

## 采集单元存储

采集单元需要三个永久性卷来存储安装文件，配置数据和日志。监控操作员使用默认存储类创建永久性卷请求。在运行安装程序脚本时，您可以使用 `-s` 选项指定其他存储类名称。

如果您的 Kubernetes 集群没有存储配置程序（例如 NetApp Trident），您可以在运行安装程序脚本时使用 `-r` 选项提供本地文件系统路径。设置 `-r` 选项后，安装程序脚本会在提供的目录中创建三个永久性卷。此目录至少需要 150 GB 的可用空间。

## 下载并运行安装脚本

Cloud Insights 提供了 Bash 脚本，用于通过监控操作员启用 Astra 数据存储预览监控。安装脚本将安装一个采集单元以及 Astra Data Store 收集器，一个 Telegraf 代理和一个 Fluent Bit 代理。

下载 Cloud Insights 租户域名和选定的 Cloud Insights API 访问令牌后，此令牌将嵌入安装程序脚本中。

然后，指标将按如下所示发送：

- Telegraf 将向 Cloud Insights 数据湖发送指标。
- Fluent Bit 会将日志发送到日志载入服务。

显示安装程序脚本帮助

安装程序脚本的完整帮助文本如下所示：

显示安装程序脚本帮助文本：

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

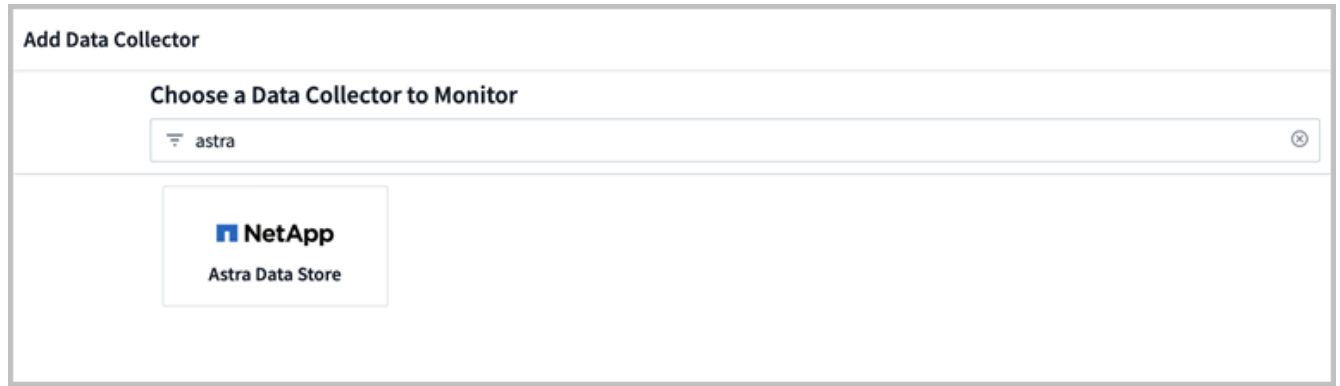
响应：

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

运行安装脚本

1. 如果您还没有 Cloud Insights 帐户，请创建一个。
2. 登录到 Cloud Insights。
3. 从 Cloud Insights 菜单中，单击 \* 管理 \* > \* 数据收集器 \*。


- 单击 \* + Data Collector\* 以添加新收集器。




- 单击 \* Astra Data Store\* 图块。
- 选择正确的 Cloud Insights API 访问令牌或创建新令牌。
- 按照说明下载安装程序脚本，更新权限并运行此脚本。

此脚本包含您的 Cloud Insights 租户 URL 和选定的 Cloud Insights API 访问令牌。






Select a Data Collector



Configure Collector




**NetApp**  
Astra Data Store

## Configure Collector

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

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

 Kubernetes

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

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

+ API Access Token

Production Best Practices ?

**Configure Astra Data Store** Need Help?

- 1

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

Copy Installer Script

☐ Reveal Installer Script

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

CI_DOMAIN_NAME="f49uaky.gstabler-ads.cloudinsights-dev.netapp.com"
CI_API_KEY="eyJraWQ1OiI5OTk5IiwidHlwIjoIc3R5bWVudDQ1fQ.eyJjcVhdG9yTG9naW41OiJhZG1pb2IiImRpc3BsYXl0YW1lIjoIZGVmYXVsdF9hZHNfYXBPX2tleTEgKG9uIGJlaGFsZiBvZiBhZG1pb2I"

```
- 3

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

Copy Permissions Command

☐ Reveal Permissions Command

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

```
- 5

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

Copy Install Command

☐ Reveal Install Command

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

```
- 7

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

Complete Setup

8. 完成脚本后，单击 \* 完成设置 \*。

安装脚本完成后，Astra Data Store 收集器将显示在 Datasources 列表中。



如果脚本因错误而退出，您可以稍后在解决错误后再次运行它。如果您的环境不使用默认设置，则该脚本支持其他参数，例如监控操作员命名空间和 Kubernetes API 服务器端口。使用 `./cloudinsights-ads-monitoring.sh -h` 中的 `-h` 选项查看使用情况和帮助文本。

如果配置成功，安装脚本将生成如下输出：

```
Configuring Cloud Insights monitoring for Astra Data Store . . .
Configuring monitoring namespace
...
Configuring output sink and Fluent Bit plugins
Configuring Telegraf plugins
Configuring Acquisition Unit
...
Acquisition Unit has been installed successfully.
Configuring Astra Data Store data collector
Astra Data Store collector data '<CLUSTER_NAME>' created
Configuration done!
```

## 代理 CR 示例

以下是运行安装程序脚本后 monitoring-NetApp 代理 CR 的外观示例。

```
spec:
  au:
    isEnabled: true
    storageClassName: auto-sc
  cluster-name: meg-ads-21-22-29-30
  docker-repo: docker.repo.eng.netapp.com/global/astra
  fluent-bit:
  - name: ads-tail
    outputs:
    - sink: ADS_STDOUT
    substitutions:
    - key: TAG
      value: firetapems
    - key: LOG_FILE
      values:
      - /var/log/firetap/*/ems/ems
      - /var/log/firetap/ems/*/ems/ems
    - key: ADS_CLUSTER_NAME
      value: meg-ads-21-22-28-29-30
  - name: agent
  - name: ads-tail-ci
    outputs:
```

```

- sink: CI
substitutions:
- key: TAG
  value: netapp.ads
- key: LOG_FILE
  values:
  - /var/log/firetap/*/ems/ems
  - /var/log/firetap/ems/*/ems/ems
- key: ADS_CLUSTER_NAME
  value: meg-ads-21-22-28-29-30
output-sink:
- api-key: abcd
  domain-name: bz19ngz.gst-adsdemo.ci-dev.netapp.com
  name: CI
serviceAccount: sa-netapp-monitoring
telegraf:
- name: ads-open-metric
  outputs:
  - sink: CI
  run-mode:
  - ReplicaSet
  substitutions:
  - key: URLS
    values:
    - http://astrads-metrics-service.astrads-
system.svc.cluster.local:9341
  - key: METRIC_TYPE
    value: ads-metric
  - key: ADS_CATEGORY
    value: netapp_ads
  - key: ADS_CLUSTER_NAME
    value: meg-ads-21-22-28-29-30
- name: agent
status:
  au-pod-status: UP
  au-uuid: eddeccc6-3aa3-4dd2-a98c-220085fae6a9

```

## 编辑 Cloud Insights 连接

您可以稍后编辑 Kubernetes API 令牌或 Cloud Insights API 访问令牌：

- 如果要更新 Kubernetes API 令牌，应从 Cloud Insights UI 编辑 Astra 数据存储收集器。
- 如果要更新用于遥测和日志的 Cloud Insights API 访问令牌，应使用 `kubectl` 命令编辑监控操作员 CR。

## 更新 **Kubernetes API** 令牌

1. 登录到 Cloud Insights 。
2. 选择 \* 管理 \* > \* 数据收集器 \* 以访问数据收集器页面。
3. 找到 Astra Data Store 集群的条目。
4. 单击页面右侧的菜单，然后选择 \* 编辑 \* 。
5. 使用新值更新 Kubernetes API Token 字段。
6. 选择 \* 保存收集器 \* 。

## 更新 **Cloud Insights API** 访问令牌

1. 登录到 Cloud Insights 。
2. 选择 \* 管理 \* > \* API 访问 \* 并单击 \* + API 访问令牌 \* ，创建新的 Cloud Insights API 访问令牌。
3. 编辑代理 CR ：

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

4. 找到 output-sink 部分，找到名为 CI 的条目。
5. 对于标签 api-key ，请将当前值替换为新的 Cloud Insights API 访问令牌。

此部分如下所示：

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

6. 保存并退出编辑器窗口。

监控操作员将更新 Telegraf 和 Fluent Bit 以使用新的 Cloud Insights API 访问令牌。

## 断开与 **Cloud Insights** 的连接

要断开与 Cloud Insights 的连接，您需要先从 Cloud Insights UI 中删除 Astra 数据存储收集器。完成后，您可以从监控操作员中删除采集单元，Telegraf 和 Fluent 位配置。

## 删除 **Astra Data Store** 预览收集器

1. 登录到 Cloud Insights 。
2. 选择 \* 管理 \* > \* 数据收集器 \* 以访问数据收集器页面。
3. 找到 Astra Data Store 集群的条目。
4. 选择屏幕右侧的菜单，然后选择 \* 删除 \* 。

5. 单击确认页面上的 \* 删除 \*。

删除采集单元， **Telegraf** 和 **Fluent** 位

1. 编辑代理 CR：

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

2. 找到 `au` 部分并将 `isenabled` 设置为 `false`
3. 找到 `flual-bit` 部分，然后删除名为 `ads-tail -CI` 的插件。如果没有其他插件，您可以删除 `flual-bit` 部分。
4. 找到 `telraf` 部分，然后删除名为 `ads-open-metric` 的插件。如果没有其他插件，您可以删除 `电报` 部分。
5. 找到 `output-sink` 部分，然后卸下名为 `CI` 的接收器。
6. 保存并退出编辑器窗口。

监控操作员将更新 **Telegraf** 和 **Fluent** 位配置并删除采集单元 POD。

7. 如果您使用本地目录作为采集单元 PV，而不是存储配置程序，请删除这些 PV：

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

然后，删除运行采集单元的节点上的实际目录。

8. 删除采集单元 POD 后，您可以从 Cloud Insights 中删除采集单元。
  - a. 在 Cloud Insights 菜单中，选择 \* 管理 \* > \* 数据收集器 \*。
  - b. 单击 \* 采集单元 \* 选项卡。
  - c. 单击采集单元 POD 旁边的菜单。
  - d. 选择 \* 删除 \*。

监控操作员将更新 **Telegraf** 和 **Fluent** 位配置并删除采集单元。

## 打开指标 **API** 帮助

下面列出了可用于从 Astra Data Store 预览版收集指标的 API。

- "help" 行说明了指标。
- "type" 行表示指标是量表还是计数器。

```
# HELP astrads_cluster_capacity_logical_percent Percentage cluster logical capacity that is used (0-100)
# TYPE astrads_cluster_capacity_logical_percent gauge
# HELP astrads_cluster_capacity_max_logical Max Logical capacity of the
```

```

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

```

```

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

```

```

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

```



```

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

```

## 配置和监控事件日志

要监控事件管理系统（EMS）日志，您可以执行以下高级任务：

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

### 在 **Astra Data Store** 预览集群自定义资源（**CR**）中配置监控

如果尚未在 Astra Data Store 预览集群 CR 上配置监控选项，您可以使用 `astrad extensions` 进行设置。

输入 ...

```

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

```

其中：

- 已安装代理的命名空间：输入监控代理的命名空间，这是监控操作员的 monitoring-NetApp CR 的默认名称。
- `-r`` 是可选的，用于设置 Fluent 或 Telegraf 映像所在的 Docker 注册表。默认情况下，此路径设置为 `docker.repo.eng.netapp.com/global/astra`，您可以进行更改。

## 设置 Cloud Insights

要查看日志，设置 Cloud Insights 是可选的；但是，使用 Cloud Insights 查看数据很有帮助。请参见 ["如何设置 NetApp Cloud Insights"](#) 用于 Astra Data Store 预览版。

## 将事件日志流式传输到 Elastic

要将 EMS 事件和其他 POD 日志流式传输到 Elastic 等第三方端点，请使用 `astrad` 扩展。

输入 ...

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



弹性主机名可以是 IP 地址。

## 将 Astra 控制中心与 Astra Data Store 预览结合使用

您可以使用 Astra 控制中心用户界面（UI）执行 Astra Data Store 预览任务。

### 为 Astra 数据存储预览设置 Astra 控制中心

要使用适用于 Astra Data Store 的 Astra 控制中心 UI 预览版，您需要完成以下任务：

- ["添加安装了 Astra Data Store 的底层 Kubernetes 集群"](#)。
- ["为该集群添加 Astra Data Store 预览存储后端"](#)。



如果您添加了存储后端，并且不存在具有 Astra Data Store 预览版的 Kubernetes 集群，则需要先添加一个集群。

### 您可以在 Astra 控制中心执行的操作

在为 Astra 数据存储预览设置 Astra 控制中心后，您可以使用 Astra 控制中心 UI 完成以下任务：

- ["使用 Astra 控制中心监控 Astra Data Store 预览资产的运行状况"](#)。
- ["管理 Astra Data Store 预览后端存储"](#)。
- ["监控节点，磁盘和永久性卷声明（PVC）"](#)。

有关详细信息 ...

- ["Astra 系列简介"](#)
- ["Astra 控制中心文档"](#)
- ["Astra Control API"](#)

## 使用自动脚本卸载 Astra Data Store 预览版

要卸载 Astra Data Store 预览和控制平面，您需要删除工作负载，绑定，卷，导出策略，Astra Data Store 集群，许可证，部署环境和 Astra Data Store 预览命名空间。

或者，您也可以 ["卸载不带脚本的 Astra Data Store 预览版"](#)。

您需要什么？ **#8217** ；将需要什么

- root 管理权限

Astra Data Store 预览卸载过程将指导您完成以下高级步骤：

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

### 删除现有工作负载和绑定

卸载 Astra Data Store 预览版之前，必须先删除以下内容

- 使用 Astra Data Store 预览版作为存储后端的所有应用程序工作负载
- 使用 Astra Data Store 预览作为后端的 Trident 绑定

这样可以确保 Kubernetes 环境保持干净，这在重新安装时非常重要。

### 卸载 Astra Data Store 集群

要卸载 Astra Data Store 预览版，您可以使用从 NetApp 支持站点下载的 Astra Data Store tar 文件中的 `uninstall.sh` 脚本。

1. 在 `manifests` 目录中找到 `uninstall.sh`。
2. 运行以下 `sed` 命令：

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

3. 运行以下脚本以指示要卸载的内容：

```
./uninstall.sh
```

You must run this script with an argument specifying what should be uninstalled

To uninstall the ADS cluster run `./uninstall.sh cluster`

To uninstall everything run `./uninstall all`

4. 如果您只想卸载集群，请输入 `uninstall.sh <cluster>`

否则，如果要卸载所有内容，请输入 `uninstall.sh`



大多数情况下，您将卸载所有内容。如果您稍后要重新部署集群，则可能只需要卸载集群。

5. 在提示符处，确认要继续并输入 `erasedata`

响应：

```
./uninstall.sh all
```

```
Enter 'erasedata' to confirm you want proceed with the uninstall:
```

```
erasedata
```

```
+-----+
```

```
| Wed Feb  2 10:14:01 EST 2022 |
```

```
| ADS cluster uninstall started |
```

```
+-----+
```

```
Deleting astradsvolumes
```

```
Deleted astradsvolumes
```

```
Deleting astradsexportpolicies
```

```
Deleted astradsexportpolicies
```

```
Deleting astradsvolumesnapshots
```

```
Deleted astradsvolumesnapshots
```

```
Deleting astradsclusters
```

```
Deleted astradsclusters
```

```
Deleting astradslicenses
```

```
Deleted astradslicenses
```

```
+-----+
```

```
| Wed Feb  2 10:15:18 EST 2022 |
```

```
| ADS cluster uninstall done |
```

```
+-----+
```

```
+-----+
```

```
| Wed Feb  2 10:15:18 EST 2022 |
```

```
| ADS system uninstall started |
```

```
+-----+
```

```
Removing astradsversion
astradsversion.astrads.netapp.io "astradsversion" deleted
Removed astradsversion
Removing daemonsets
daemonset.apps "astrads-ds-nodeinfo-astradsversion" deleted
Removed daemonsets
Removing deployments
deployment.apps "astrads-cluster-controller" deleted
deployment.apps "astrads-license-controller" deleted
deployment.apps "astrads-operator" deleted
Removed deployments
Removing all other AstraDS resources
namespace "astrads-system" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsautosupports.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradscLOUDsnapshots.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsclusters.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsexportpolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsfaileddrives.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradslicenses.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnfsOPTIONS.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnODEinfoes.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsnODEmanagements.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsqOSPolicies.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsversions.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMEfiles.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMES.astrads.netapp.io" deleted
customresourcedefinition.apiextensions.k8s.io
"astradsvOLUMESnapshots.astrads.netapp.io" deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-admin-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-reader-role"
deleted
role.rbac.authorization.k8s.io "astrads-astrads-system-writer-role"
```

```
deleted
role.rbac.authorization.k8s.io "astrads-leader-election-role" deleted
role.rbac.authorization.k8s.io "astrads-manager-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-admin-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-reader-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astrads-writer-
clusterrole" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsautosupport-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscloudsnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradscluster-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsexportpolicy-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsfaileddrive-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradslicense-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnfsoption-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodeinfo-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsnodemanagement-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsqospolicy-viewer-
```

```

role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsversion-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolume-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-editor-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumefile-viewer-
role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
editor-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-astradsvolumesnapshot-
viewer-role" deleted
clusterrole.rbac.authorization.k8s.io "astrads-manager-role" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-leader-election-
rolebinding" deleted
rolebinding.rbac.authorization.k8s.io "astrads-manager-rolebinding"
deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-admin-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-reader-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-astrads-writer-
rolebinding" deleted
clusterrolebinding.rbac.authorization.k8s.io "astrads-manager-
rolebinding" deleted
configmap "astrads-autosupport-cm" deleted
configmap "astrads-firetap-cm" deleted
configmap "astrads-kevents-asup" deleted
configmap "astrads-metrics-cm" deleted
secret "astrads-autosupport-certs" deleted
+-----+
| Wed Feb  2 10:16:36 EST 2022                |
| ADS system uninstall done                    |
+-----+

```

## 验证是否删除了 **astrads-system** 命名空间

确保以下命令不返回任何结果：

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

## 确保工作节点上未运行容器

验证 **fireap** 或 **netwd** 等容器是否未在工作节点上运行。在每个节点上运行以下命令。

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

## 删除 **OpenShift** 容器平台资源

如果您在 Red Hat OpenShift 容器平台（OCP）上安装了 Astra Data Store preview，则可以卸载 OCP 安全上下文约束（SCC）和绑定资源。

OpenShift 使用安全上下文约束（SCC）来控制 Pod 可以执行的操作。

完成标准卸载过程后，请完成以下步骤。

1. 删除 SCC 资源：

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

2. 删除 rolebindings 资源：

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



忽略这些步骤中的 " 未找到资源 " 错误。

3. 从所有 Kubernetes 节点中删除 `/var/lib/kubelet/config.yaml`。

## 对 **Astra Data Store** 预览卸载过程进行故障排除

Kubernetes v1.20 中的 Astra 数据存储预览卸载过程有时可能会使发生原因 Pod 保持终止状态。

如果发生此问题描述，请运行以下命令强制删除 **astrads-system** 命名空间中的所有 Pod：

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



# 卸载不带脚本的 Astra Data Store 预览版

要在不使用自动脚本的情况下手动卸载 Astra Data Store 预览版，您需要删除工作负载，绑定，卷，导出策略，集群，许可证，部署环境和 Astra Data Store 预览命名空间。

或者，您也可以 ["使用脚本卸载 Astra Data Store 预览版"](#)。

您需要什么？ **#8217** ；将需要什么

- root 管理权限

Astra Data Store 预览卸载过程将指导您完成以下高级步骤：

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

## 删除现有工作负载和绑定

卸载 Astra Data Store 预览版之前，必须先删除以下内容

- 使用 Astra Data Store 预览版作为存储后端的所有应用程序工作负载
- 使用 Astra Data Store 预览作为后端的 Trident 绑定

这样可以确保 Kubernetes 环境保持干净，这在重新安装时非常重要。

## 卸载 Astra Data Store 预览集群和控制平面

按照以下步骤手动卸载 Astra Data Store 预览版。

### 删除卷和导出策略

在删除集群之前，您应删除 Astra Data Store 预览卷和导出策略。



如果不先删除卷和导出策略，则集群删除过程将暂停，直到删除 Astra Data Store 预览卷对象为止。在开始删除集群之前删除这些项会更高效。

### 步骤

1. 删除卷：

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

## 2. 删除导出策略：

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

## 删除 Astra Data Store 预览集群

删除集群时，只会删除 Astra Data Store 预览集群对象自定义资源（CR）以及集群范围的资源。



即使在删除集群后，操作符，nodeinfo Pod 和集群控制器（即 Kubernetes 范围的资源）仍会保持不变。

删除集群还会从节点卸载底层操作系统，从而停止 fireap 和 netwd 服务。

卸载程序大约需要一分钟才能完成。然后，开始删除 Astra Data Store 预览集群范围的资源。

## 1. 删除集群：

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

## 删除许可证

1. 通过 SSH 连接到集群中的每个工作节点，并验证`fireap`或`netwd`未在工作节点中运行。
2. 删除 Astra Data Store 预览许可证：

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

## 删除 Astra Data Store 预览安装

删除集群中的控制器，操作员，命名空间和支持 Pod。

## 1. 删除 Astra Data Store 预览安装对象：

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

## 2. 删除数据存储 DemonSets 和所有 Astra Data Store 预览控制器资源：

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

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

## 3. 删除其余项目和操作符 YAML 文件：

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

## 验证是否删除了 **astrads-system** 命名空间

确保以下命令不返回任何结果：

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

## 确保工作节点上未运行容器

验证 **fireap** 或 **netwd** 等容器是否未在工作节点上运行。在每个节点上运行以下命令。

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

## 删除 **OpenShift** 容器平台资源

如果您在 Red Hat OpenShift 容器平台（OCP）上安装了 Astra Data Store preview，则可以卸载 OCP 安全上下文约束（SCC）和绑定资源。

OpenShift 使用安全上下文约束（SCC）来控制 Pod 可以执行的操作。

完成标准卸载过程后，请完成以下步骤。

### 1. 删除 SCC 资源：

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

### 2. 删除 rolebindings 资源：

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



忽略这些步骤中的 " 未找到资源错误 "。

3. 从所有 Kubernetes 节点中删除 `/var/lib/kubelet/config.yaml`。

## 手动删除示例

下面显示了执行手动卸载脚本的示例。

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

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

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

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

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

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

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

```

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

$ kubectl get ns | grep astrads-system

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

```

## 对 Astra Data Store 预览卸载过程进行故障排除

Kubernetes v1.20 中的 Astra 数据存储预览卸载过程有时可能会使发生原因 Pod 保持终止状态。

如果发生此问题描述，请运行以下命令强制删除 astrads-system 命名空间中的所有 Pod：

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

# 知识和支持

## 故障排除

了解如何解决您可能遇到的一些常见问题。

[https://kb.netapp.com/Advice\\_and\\_Troubleshooting/Cloud\\_Services/Astra](https://kb.netapp.com/Advice_and_Troubleshooting/Cloud_Services/Astra)

## 获取帮助

NetApp 通过多种方式为 Astra Data Store 预览版提供支持。全天候提供免费自助服务支持选项，例如知识库（KB）文章和可宽延的渠道。



您可以获得有关 Astra Data Store 预览版的社区技术支持。使用创建案例 "[NetApp 支持站点（NSS）](#)" 不适用于预览版本。您可以通过反馈选项联系支持部门，也可以使用 Slack 渠道自助服务。

### 自助服务支持选项

这些选项全天候免费提供：

- "[知识库（需要登录）](#)"

搜索与 Astra Data Store 预览相关的文章，常见问题解答或中断修复信息。

- 文档。

这是您当前正在查看的文档站点。

- "[netapp](#)" 容器 "可宽延通道"

转到 "容器" 渠道，与同行和专家建立联系。

- 反馈电子邮件

发送电子邮件至 [astra.feedback@netapp.com](mailto:astra.feedback@netapp.com)，告知我们您的想法，想法或顾虑。

### 了解更多信息

- "[如何将文件上传到 NetApp（需要登录）](#)"
- "[NetApp 知识库文章](#)"

## 自动支持监控

AutoSupport 监控 Astra 数据存储预览系统运行时间和信息，并向 NetApp 支持部门发送消息。根据您的配置，可以监控以下系统组件：

- 控制面板
- 存储

默认情况下，期间会启用 AutoSupport ["Astra Data Store 预览集群安装"](#) 或者在将 AutoSupport CR 应用于集群后。启用后，AutoSupport（ASUP）捆绑包将自动上传到 ["NetApp 支持站点（NSS）"](#) 或可供手动下载。

#### 选项

- [\[AutoSupport triggers and scenarios\]](#)
- [\[Configure custom control plane AutoSupport collection\]](#)
- [\[Configure custom storage AutoSupport collection\]](#)
- [\[List ASUPs in the system\]](#)
- [\[Download an ASUP Bundle\]](#)
- [\[Upload a core file\]](#)

## AutoSupport 触发器和场景

AutoSupport 捆绑包可通过以下方式触发：

- 定期：ASUP 捆绑包按 CR 中定义的间隔创建。
- User triggered：您可以手动创建自己的 ASUP 以查看日志。
- 核心转储：如果节点上存在核心转储，则会生成 ASUP，并将核心发送给 NetApp 进行进一步调查。
- 基于 CallHome 事件：从操作系统的特定 CallHome 事件生成 ASUP。
- 基于 Kubernetes 事件：通过控制平面中的特定 Kubernetes 事件生成 ASUP。

这些触发情形会生成以下 AutoSupport 类型之一：

- ControlPlane AutoSupport：一组 Astra 数据存储预览控制平面日志和 CRS。
- Storage AutoSupport：存储报告和性能数据的集合。
- 核心转储 AutoSupport：系统核心转储的集合。

## 配置自定义控制平面 AutoSupport 收集

您可以创建一个自定义 AutoSupport 收集配置，用于报告控制平面事件。默认情况下，大多数安装都已在期间启用定期事件报告 ["Astra Data Store 预览集群安装"](#)。此操作步骤介绍如何配置 AutoSupport CR，以便根据您的参数进行报告：

#### 步骤

1. 自定义以下命令以创建控制面板集合 CR：

```
kubectl astasds asup collect -c controlplane --namespace=astrads-system
```

- a. 定义自定义参数：

- `<myASUPname>`：要生成的 AutoSupport CR 的名称。



- `-e <事件名称>`：触发收集事件名称的事件名称应在 component-YAML（挂载到支持控制器）中预定义。

#### 示例

```
kubectl astrasds asup collect -c controlplane custom-asup-name -e debug
--namespace=astrads-system
```

#### a. 根据需要为系统添加其他参数：

- `-cluster`：在多集群环境中需要此标志。
- `-localCollection`：启用本地收集。默认值为 `false`。
- `-forceUpload`：启用强制上传。默认值为 `false`。
- `--retry`：启用重试。默认值为 `false`。

## 配置自定义 **Storage AutoSupport** 收集

您可以创建自定义 AutoSupport 收集配置，以报告存储组件事件。默认情况下，大多数安装都已在期间启用定期事件报告 "[Astra Data Store 预览集群安装](#)"。此操作步骤介绍如何配置 AutoSupport CR，以便根据您选择的参数进行报告：

#### 步骤

#### 1. 自定义以下命令以创建存储收集 CR：

```
kubectl astrasds asup collect -c storage --namespace=astrads-system
```

#### a. 定义自定义参数：

- `<myASUPname>`：要生成的 AutoSupport CR 的名称。
- `-e <事件名称>`：触发收集事件名称的事件名称应在 component-YAML（挂载到支持控制器）中预定义。

#### 性能事件示例：

```
kubectl astrasds asup collect -c storage -e performance example-perf-storage-asup
```

- `-t <ISO_format> -d <hours>`：在指定的持续时间内为所有节点收集存储 ASUP。请使用标准 ISO 日期时间格式（`-t`），持续时间（`D`）以小时为单位。例如：

```
kubectl astrasds asup collect -c storage -t 2021-01-01T15:00:00Z -d 24
```

- `-nodes <节点名称>`：为指定节点收集存储 ASUP。例如：

```
kubectl astrads asup collect -c storage --nodes example1
```

- `-nodes nodename1, nodeame2, nodeame3`：为指定节点收集存储 ASUP：

```
kubectl astrads asup collect -c storage --nodes  
example1,example2,example3
```

a. 根据需要为系统添加其他参数：

- `-cluster`：在多集群环境中需要此标志。
- `-localCollection`：启用本地收集。默认值为 `false`。
- `-forceUpload`：启用强制上传。默认值为 `false`。
- `--retry`：启用重试。默认值为 `false`。

## 列出系统中的 ASUP

使用以下命令按名称列出系统中的 ASUP：

```
kubectl astrads asup list --namespace=astrads-system
```

响应示例：

NAMESPACE	NAME	SEQUENCE	NUMBER	EVENT
SIZE	STATE	LOCAL	COLLECTION	
astrads-system	storage-callhome.reboot.unknown-...	1		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-tdl2h:	
astrads-system	storage-callhome.reboot.unknown-...	2		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-xx6n8:	
astrads-system	storage-callhome.reboot.unknown-...	3		
callhome.reboot.unknown	0	uploaded	astrads-ds-support-qghnx:	

## 下载 ASUP 捆绑包

您可以使用此命令下载本地收集的 ASUP 捆绑包。使用 `-o <位置>` 指定当前工作目录以外的位置：

```
./kubectl-astrasds asup download <ASUP_bundle_name> -o <location>
```

## 上传核心文件

如果服务崩溃，则会创建 AutoSupport（ASUP）消息以及崩溃时包含相关内存内容的文件（称为核心文件）。Astra Data Store 预览版会自动将 ASUP 消息上传到 NetApp 支持，但您需要手动上传核心文件，以便它与

ASUP 消息关联。

步骤

1. 使用以下 `kubectl` 命令查看 ASUP 消息：

```
kubectl astrads asup list --namespace=astrads-system
```

您应看到类似于以下内容的输出：

NAMESPACE	NAME	SEQUENCE NUMBER	EVENT
SIZE	STATE	LOCAL COLLECTION	
astrads-system	storage-coredump-2021...	1	coredump
197848373	compressed	astrads-ds-support-sxxn7:/var/...	

2. 使用以下 `kubectl` 命令从 ASUP 消息中下载核心文件。使用 `-o`` 选项为下载的文件指定目标目录。

```
kubectl astrads asup download storage-coredump-20211216t140851311961680  
-o <absolute_path_to_destination_directory>
```



在极少数情况下，您可能无法下载核心文件，因为其他核心文件已取代它。发生这种情况时，命令将返回错误 `cannot stat : no such file or directory`。如果您看到此错误，可以 ["获取帮助"](#)。

3. 打开 Web 浏览器并浏览到 ["NetApp 身份验证文件上传工具"](#)，如果您尚未登录，请输入您的 NetApp 支持凭据。
4. 选中 `* 我没有案例编号 *` 复选框。
5. 在 `* 最近的区域 *` 菜单中，选择最接近您的区域。
6. 选择 `* 上传 *` 按钮。
7. 浏览并选择先前下载的核心文件。

此时将开始上传。上传完成后，将显示一条成功消息。

了解更多信息

- ["如何将文件上传到 NetApp（需要登录）"](#)

# 法律声明

""

""

"Astra Data Store 21.12 版本的通知"

## Copyright Information

Copyright © 2022 NetApp, Inc. All rights reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means-graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system-without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

## Trademark Information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.