入门

Astra Control Center

NetApp
August 11, 2025

目录

入门

了解Astra Control

Astra Control 是 Kubernetes 应用程序数据生命周期管理解决方案，可简化有状态应用程序的操作。轻松保护、备份、复制和迁移Kubernetes工作负载、并即时创建有效的应用程序克隆。

功能

Astra Control 为 Kubernetes 应用程序数据生命周期管理提供了关键功能：

- 自动管理永久性存储
- 创建应用程序感知型按需快照和备份
- 自动执行策略驱动的快照和备份操作
- 将应用程序和数据从一个 Kubernetes 集群迁移到另一个集群
- 使用NetApp SnapMirror技术(Astra Control Center)将应用程序复制到远程系统
- 将应用程序从暂存克隆到生产
- 直观显示应用程序运行状况和保护状态
- 使用Web UI或API实施备份和迁移工作流

部署模式

Astra Control 有两种部署模式：

- * Astra Control Service*：NetApp管理的服务、可为多个云提供商环境中的Kubernetes集群以及自管理Kubernetes集群提供应用程序感知型数据管理。
- * Astra Control Center* ：自管理软件，可为内部环境中运行的 Kubernetes 集群提供应用程序感知型数据管理。Astra控制中心还可以安装在多个云提供商环境中、并具有一个NetApp Cloud Volumes ONTAP 存储后端。

	Astra 控制服务	**Astra** 控制中心
如何提供?	作为 NetApp 提供的一项完全托管的云服务	作为可下载、安装和管理的软件
它托管在何处?	基于 NetApp 选择的公有云	在您自己的Kubernetes集群上
如何更新?	由 NetApp 管理	您可以管理任何更新

	Astra 控制服务	**Astra** 控制中心
支持哪些**Kubednetes**分发版?	• 云提供商 ◦ Amazon Web Services ▪ Amazon Elelic Kubelnetes Service (EKS) ◦ Google Cloud ▪ Google Kubernetes Engine （ GKEE ） ◦ Microsoft Azure ▪ Azure Kubernetes Service （ AKS ） • 自管理集群 ◦ Kubnetes (上游) ◦ Rancher Kubernetes Engine （ RKE） ◦ Red Hat OpenShift 容器平台 • 内部集群 ◦ Red Hat OpenShift容器平台内部部署	• 基于Azure堆栈HCI的Azure Kubnetes Service • Google Anthos • Kubnetes (上游) • Rancher Kubernetes Engine （ RKE） • Red Hat OpenShift 容器平台

	Astra 控制服务	Astra 控制中心
支持哪些存储后端?	• 云提供商 ◦ Amazon Web Services ▪ Amazon EBS ▪ 适用于 NetApp ONTAP 的 Amazon FSX ▪ "Cloud Volumes ONTAP" ◦ Google Cloud ▪ Google 持久磁盘 ▪ NetApp Cloud Volumes Service ▪ "Cloud Volumes ONTAP" ◦ Microsoft Azure ▪ Azure受管磁盘 ▪ Azure NetApp Files ▪ "Cloud Volumes ONTAP" • 自管理集群 ◦ Amazon EBS ◦ Azure受管磁盘 ◦ Google 持久磁盘 ◦ "Cloud Volumes ONTAP" ◦ NetApp MetroCluster ◦ "Longhorn" • 内部集群 ◦ NetApp MetroCluster ◦ NetApp ONTAP AFF 和 FAS 系统 ◦ NetApp ONTAP Select ◦ "Cloud Volumes ONTAP" ◦ "Longhorn"	• NetApp ONTAP AFF 和 FAS 系统 • NetApp ONTAP Select • "Cloud Volumes ONTAP" • "Longhorn"

Astra 控制服务的工作原理

Astra Control Service 是一种由 NetApp 管理的云服务，它始终处于启用状态，并使用最新功能进行更新。它利用多个组件实现应用程序数据生命周期管理。

从较高的层面来看， Astra Control Service 的工作原理如下：

- 您可以通过设置云提供商并注册 Astra 帐户开始使用 Astra Control Service 。

- 对于 GKE- 集群， Astra Control Service 使用 "适用于 Google Cloud 的 NetApp Cloud Volumes Service" 或 Google Persistent Disk 作为永久性卷的存储后端。
- 对于 AKS 集群， Astra Control Service 使用 "Azure NetApp Files" 或Azure受管磁盘作为永久性卷的存储后端。
- 对于Amazon EKS集群、Astra Control Service使用 "Amazon Elastic Block Store" 或 "适用于 NetApp ONTAP 的 Amazon FSX" 作为永久性卷的存储后端。

- 您可以将第一个 Kubernetes 计算添加到 Astra Control Service 中。然后， Astra 控制服务将执行以下操作：
  - 在云提供商帐户中创建一个对象存储，该帐户是备份副本的存储位置。

    在 Azure 中， Astra Control Service 还会为 Blob 容器创建资源组，存储帐户和密钥。
  - 在集群上创建新的管理员角色和 Kubernetes 服务帐户。
  - 使用此新管理员角色在集群上安装link./概念/architution#Astra-control-components [Astra Control置备程序^]、并创建一个或多个存储类。
  - 如果您使用NetApp云服务存储产品作为存储后端、Astra控制服务将使用Astra控制配置程序为应用程序配置永久性卷。如果您使用Amazon EBS或Azure托管磁盘作为存储后端、则需要安装特定于提供商的CSI驱动程序。中提供了安装说明 "设置Amazon Web Services" 和 "使用 Azure 受管磁盘设置 Microsoft Azure"。
- 此时，您可以向集群添加应用程序。将在新的默认存储类上配置永久性卷。
- 然后，您可以使用 Astra Control Service 管理这些应用程序，并开始创建快照，备份和克隆。

Astra Control的免费计划支持您管理帐户中多达10个命名空间。如果您要管理10个以上的计划、则需要通过从"免费计划"升级到"高级计划"来设置计费。

Astra 控制中心的工作原理

Astra 控制中心在您自己的私有云中本地运行。

Astra控制中心支持Kuburenet集群、其中Astra控制配置程序配置了存储类、并具有ONTAP存储后端。

Astra Control Center提供有限的(7天的指标)监控和遥测功能、还可通过开放式指标端点导出到Kubarnetes本机监控工具(如Prometheus和Grafana)。

Astra控制中心完全集成到AutoSupport和Active IQ数字顾问(也称为数字顾问)生态系统中、可为用户和NetApp支持提供故障排除和使用信息。

您可以使用90天嵌入式评估许可证试用Astra Control Center。在评估Astra Control Center时、您可以通过电子邮件和社区选项获得支持。此外，您还可以从产品支持信息板访问知识库文章和文档。

要安装和使用 Astra 控制中心，您需要满足特定的要求 "要求"。

从较高的层面来看， Astra 控制中心的工作原理如下：

- 您可以在本地环境中安装 Astra Control Center 。详细了解如何操作 "安装 Astra 控制中心"。
- 您可以完成一些设置任务，例如：
  - 设置许可

- 添加第一个集群。
- 添加在添加集群时发现的存储后端。
- 添加用于存储应用程序备份的对象存储分段。

详细了解如何操作 "设置 Astra 控制中心"。

您可以将应用程序添加到集群中。或者、如果要管理的集群中已有一些应用程序、则可以使用Astra控制中心对其进行管理。然后、使用Astra控制中心创建快照、备份、克隆和复制关系。

有关详细信息 ...

- "Astra Control Service 文档"
- "Astra 控制中心文档"
- "Astra Trident 文档"
- "Astra Control API文档"
- "ONTAP 文档"

Astra 控制中心要求

首先验证操作环境，应用程序集群，应用程序，许可证和 Web 浏览器的就绪情况。确保您的环境满足这些要求、以部署和运行Astra Control Center。

支持的主机集群**Kubennetes**环境

Astra Control Center已通过以下Kubenetes主机环境的验证：

确保您选择托管Astra Control Center的Kubennet环境满足环境官方文档中列出的基本资源要求。

主机集群上的**Kubnetes**分发	支持的版本
基于Azure堆栈HCI的Azure Kubnetes Service	采用AKS 1.24.11至1.26.6的Azure Stack HCI 21H2和22H2
Google Anthos	1.15至1.16 (请参见 Google Anthos入口要求)
Kubnetes (上游)	1.27至1.29
Rancher Kubernetes Engine （ RKE）	RKE 1：版本1.24.17、1.25.13、1.26.8、带RANcher Manager 2.7.9 RKE 2：版本1.23.16和1.24.13、带Randcher Manager 2.6.13 RKE 2：版本1.24.17、1.25.14、1.26.9、带RANcher Manager 2.7.9
Red Hat OpenShift 容器平台	4.12至4.14

主机集群资源要求

除了环境的资源要求之外， Astra 控制中心还需要以下资源：

> 这些要求假定 Astra 控制中心是运行环境中唯一运行的应用程序。如果环境运行的是其他应用程序，请相应地调整这些最低要求。

- **CPU**扩展：托管环境中所有节点的CPU都必须启用AVX扩展。
- 工作节点：总共至少3个工作节点、每个节点具有4个CPU核心和12 GB RAM
- **VMware Tanzu Kubernetes Grid**集群要求：在VMware Tanzu Kubernetes Grid (TKG)或Tanzu Kubernetes Grid Integrated Edition (TKGi)集群上托管Astra Control Center时，请记住以下注意事项。
  - 默认的 VMware TKG 和 TKGi 配置文件令牌将在部署后 10 小时过期。如果您使用的是 Tanzu 产品组合，则必须使用未过期的令牌生成 Tanzu Kubernetes 集群配置文件，以防止 Astra 控制中心与受管应用程序集群之间出现连接问题。有关说明，请访问 "VMware NSX-T 数据中心产品文档。"
  - 使用 `kubectl get nsxlbmonitors -A` 命令以查看是否已将服务监控器配置为接受传入流量。如果存在一个，则不应安装 MetalLB ，因为现有服务监控器将覆盖任何新的负载平衡器配置。
  - 在任何要由 Astra Control 管理的应用程序集群上禁用 TKG 或 TKGi 默认存储类强制实施。您可以通过编辑来执行此操作 `TanzuKubernetesCluster` 命名空间集群上的资源。
  - 在TKG或TKGi环境中部署Astra Control Center时、请注意Astra Control配置程序的特定要求：
    - 集群必须支持有权限的工作负载。
    - 。 `--kubelet-dir` 标志应设置为kubelet目录的位置。默认情况下、此值为 `/var/vcap/data/kubelet`。
    - 使用指定kubelet位置 `--kubelet-dir` 已知适用于Trident操作员、Helm和 `tridentctl` 部署。

服务网格要求

强烈建议您在Astra Control Center主机集群上安装受支持的vilano版本的Istio服务网格。请参见 "支持的版本" 支持的伊斯托伊奥版本。Itio服务网格的品牌版本(例如OpenShift Service Mesh)未通过Astra Control Center的验证。

要将Astra Control Center与主机集群上安装的Isio服务网格集成、必须将此集成作为Astra Control Center的一部分 "安装" 而不是独立于此过程。

> 如果在主机集群上安装和使用Astra Control Center而不配置服务网格、则可能会产生严重的安全影响。

Astra Trident

如果您要在此版本中使用Asta三端安装程序而不是Asta Control配置程序、则支持Asta三端安装程序23.04及更高版本。Asta Control Center需要 Asta Control配置程序 在未来版本中。

Asta Control配置程序

要使用Astra Control配置程序高级存储功能、必须安装Astra Trident 23.10或更高版本并启用 "Astra Control配置程序功能"。要使用最新的Astra Control配置程序功能、您需要最新版本的Astra三端和Astra Control Center。

- 与**Asta Control Center**一起使用的**Asta Control**配置程序的最低版本：已安装并配置Asta Control配置程序23.10或更高版本。

采用**Asta**三端磁盘的**ONTAP**配置

- 存储类：在集群上至少配置一个存储类。如果配置了默认存储类、请确保它是唯一具有默认指定的存储类。
- 存储驱动程序和工作节点：确保为集群中的工作节点配置适当的存储驱动程序，以便Pod可以与后端存储进行交互。Astra 控制中心支持由 Astra Trident 提供的以下 ONTAP 驱动程序：
  - `ontap-nas`
  - `ontap-san`
  - `ontap-san-economy` (此存储类类型不支持应用程序复制)
  - `ontap-nas-economy` (快照和应用程序复制策略不适用于此存储类类型)

存储后端

请确保您有一个受支持的后端、并具有足够的容量。

- 所需存储后端容量：至少500 GB可用
- 支持的后端：Astra Control Center支持以下存储后端：
  - NetApp ONTAP 9.9.1或更高版本的AFF、FAS和ASA系统
  - NetApp ONTAP Select 9.9.1或更高版本
  - NetApp Cloud Volumes ONTAP 9.9.1或更高版本
  - (适用于Astra控制中心技术预览版) NetApp ONTAP 9.10.1或更高版本、用于数据保护操作、作为技术预览版提供
  - Longhorn 1.5.0或更高版本
    - 需要手动创建卷SnapshotClass对象。请参见 "Longhorn文档" 有关说明，请参见。
  - NetApp MetroCluster
    - 受管Kubbernetes集群必须采用延伸型配置。
  - 支持的云提供商提供存储后端

ONTAP 许可证

要使用Astra控制中心、请根据您需要完成的任务、验证您是否具有以下ONTAP 许可证：

- FlexClone
- SnapMirror：可选。只有在使用SnapMirror技术复制到远程系统时才需要。请参见 "SnapMirror许可证信息"。
- S3许可证：可选。只有ONTAP S3存储分段才需要

要检查ONTAP 系统是否具有所需的许可证、请参见 "管理ONTAP 许可证"。

NetApp MetroCluster

使用NetApp MetroCluster作为存储后端时、您需要执行以下操作：

- 在您使用的Asta三端驱动程序中、将SVM管理LIF指定为后端选项
- 确保您拥有相应的ONTAP许可证

要配置MetroCluster LIF、请参阅每个驱动程序的以下选项和示例：

- "SAN"
- "NAS"

Asta Control Center许可证

Astra Control Center需要Astra Control Center许可证。安装Astra Control Center时、已激活4、800个CPU单元的嵌入式90天评估版许可证。如果您需要更多容量或不同的评估条款、或者要升级到完整许可证、则可以从NetApp获得不同的评估许可证或完整许可证。您需要一个许可证来保护应用程序和数据。

您可以通过注册获取免费试用版来试用Astra Control Center。您可以通过注册进行注册 "此处"。

要设置许可证、请参见 "使用 90 天评估许可证"。

要了解有关许可证工作原理的详细信息、请参见 "许可"。

网络要求

配置操作环境以确保Astra Control Center可以正确通信。需要以下网络配置：

- **FQDN**地址:您必须拥有Astra Control Center的FQDN地址。
- 访问互联网：您应确定是否可以从外部访问互联网。否则、某些功能可能会受到限制、例如向发送支持包 "NetApp 支持站点"。
- 端口访问：Astra Control Center的运行环境使用以下TCP端口进行通信。您应确保允许这些端口通过任何防火墙，并将防火墙配置为允许来自 Astra 网络的任何 HTTPS 传出流量。某些端口需要在托管 Astra 控制中心的环境与每个受管集群之间进行双向连接（请在适用时注明）。

> 您可以在双堆栈 Kubernetes 集群中部署 Astra 控制中心，而 Astra 控制中心则可以管理为双堆栈操作配置的应用程序和存储后端。有关双堆栈集群要求的详细信息，请参见 "Kubernetes 文档"。

源	目标	**Port**	协议	目的
客户端 PC	Astra 控制中心	443.	HTTPS	UI/API访问-确保Astra控制中心与用于访问Astra控制中心的系统之间的双向端口处于打开状态
指标使用者	Astra 控制中心工作节点	9090	HTTPS	指标数据通信—确保每个受管集群都可以访问托管 Astra 控制中心的集群上的此端口 （需要双向通信）

源	目标	Port	协议	目的
Astra 控制中心	Amazon S3 存储分段提供商	443.	HTTPS	Amazon S3 存储通信
Astra 控制中心	NetApp AutoSupport	443.	HTTPS	NetApp AutoSupport 通信
Astra 控制中心	托管Kubbernetes集群	443/6443 **NOTE**：受管集群使用的端口可能因集群而异。请参见集群软件供应商提供的文档。	HTTPS	与受管集群通信-确保托管Asta Control Center的集群与每个受管集群之间的此端口均处于打开状态

内部 **Kubernetes** 集群的传入

您可以选择 Astra 控制中心使用的网络传入类型。默认情况下， Astra 控制中心会将 Astra 控制中心网关（ service/traefik ）部署为集群范围的资源。如果您的环境允许使用服务负载平衡器，则 Astra 控制中心也支持使用服务负载平衡器。如果您希望使用服务负载平衡器、但尚未配置此平衡器、则可以使用MetalLB负载平衡器自动为该服务分配外部IP地址。在内部 DNS 服务器配置中，您应将为 Astra 控制中心选择的 DNS 名称指向负载平衡的 IP 地址。

负载平衡器应使用与Astra控制中心工作节点IP地址位于同一子网中的IP地址。

有关详细信息，请参见 "设置传入以进行负载平衡"。

Google Anthos入口要求

如果在Google Anthos集群上托管Astra Control Center、请注意、默认情况下、Google Anthos包括MetalLB负载平衡器和Istio入口服务、您只需在安装期间使用Astra Control Center的通用入口功能即可。请参见 "Astra Control Center安装文档" 了解详细信息。

支持的 **Web** 浏览器

Astra 控制中心支持最新版本的 Firefox， Safari 和 Chrome ，最小分辨率为 1280 x 720 。

应用程序集群的其他要求

如果您计划使用以下Astra控制中心功能、请记住这些要求：

- 应用程序集群要求： "集群管理要求"
  - 受管应用程序要求： "应用程序管理要求"
  - 应用程序复制的其他要求： "复制前提条件"

下一步行动

查看 "快速入门" 概述。

Astra 控制中心快速入门

下面简要介绍了开始使用Astra控制中心所需的步骤。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。

查看 **Kubernetes** 集群要求

确保您的环境满足以下要求：

- Kubernetes集群*
- "确保主机集群满足操作环境要求"
- "为内部Kubernetes集群的负载平衡配置传入"

存储集成

- "确保您的环境包含Astra Control配置程序"
- "启用Astra Control配置程序高级管理和存储配置功能"
- "准备集群工作节点"
- "配置存储后端"
- "配置存储类"
- "安装卷快照控制器"
- "创建卷快照类"
- ONTAP 凭据*
- "配置ONTAP 凭据"

下载并安装**Astra**控制中心

完成以下安装任务：

- "从NetApp 支持站点 下载页面下载Astra控制中心"
- 获取NetApp许可证文件：
  - 如果您正在评估Astra Control Center、则已包含嵌入式评估许可证
  - "如果您已购买Astra Control Center、请生成许可证文件"
- "安装 Astra 控制中心"
- "执行其他可选配置步骤"

完成一些初始设置任务

完成一些基本任务以开始使用：

- "添加许可证"

- "准备用于集群管理的环境"
- "添加集群"
- "添加存储后端"
- "添加存储分段"

4 使用 **Astra** 控制中心

完成Asta Control Center设置后、请使用Asta Control UI或 "Astra Control API" 要开始管理和保护应用程序、请执行以下操作：

- "管理帐户"：用户、角色、LDAP、凭据等。
- "管理通知"
- "管理应用程序"：定义要管理的资源。
- "保护应用程序"：配置保护策略以及复制、克隆和迁移应用程序。

有关详细信息 ...

- "使用 Astra Control API"
- "升级 Astra 控制中心"
- "获取有关Astra Control的帮助"

安装概述

选择并完成以下 Astra 控制中心安装过程之一：

- "使用标准流程安装 Astra 控制中心"
- "（如果使用 Red Hat OpenShift ）使用 OpenShift OperatorHub 安装 Astra 控制中心"
- "使用 Cloud Volumes ONTAP 存储后端安装 Astra 控制中心"

根据您的环境、安装Astra控制中心后可能需要进行其他配置：

- "安装后配置Astra控制中心"

使用标准流程安装 **Astra** 控制中心

要安装Astra Control Center、请下载安装映像并执行以下步骤。您可以使用此操作步骤在互联网连接或通风环境中安装 Astra 控制中心。

有关Asta Control Center安装过程的演示，请参见 "此视频"。

开始之前

- 满足环境前提条件： "开始安装之前，请为 Astra Control Center 部署准备您的环境"。

在第三个容错域或二级站点中部署A作用 力控制中心。对于应用程序复制和无缝灾难恢复、建议执行此操作。

- 确保服务运行状况良好：检查所有API服务是否均处于运行状况良好且可用：

```
kubectl get apiservices
```

- 确保具有可路由的**FQDN**：您计划使用的Astra FQDN可以路由到集群。这意味着您的内部 DNS 服务器中有一个 DNS 条目，或者您正在使用已注册的核心 URL 路由。
- 配置证书管理器：如果集群中已存在证书管理器，则需要执行某些操作 "前提条件步骤" 这样、Astra控制中心就不会尝试安装自己的证书管理器。默认情况下、Astra控制中心会在安装期间安装自己的证书管理器。
- **(仅限ONTAP SAN**驱动程序)启用多路径：如果使用的是ONTAP SAN驱动程序、请确保在所有Kubernetes集群上启用了多路径。

您还应考虑以下事项：

- 获取**NetApp Astra**控件映像注册表的访问权限：

  您可以选择从NetApp映像注册表中获取Astra控件的安装映像和增强功能、例如Astra控件配置程序。

  a. 记录您登录注册表所需的Astra Control帐户ID。

     您可以在Astra Control Service Web UI中查看您的帐户ID。选择页面右上角的图图标，选择*API access*并记下您的帐户ID。

  b. 在同一页面中，选择*Generate API t令牌*并将API令牌字符串复制到剪贴板，然后将其保存在编辑器中。

  c. 登录到Asta Control注册表：

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- 安装用于安全通信的服务网格：强烈建议使用保护Astra Control主机集群通信通道的安全 "支持的服务网格"。

只能在Astra Control Center期间将Astra Control Center与服务网格集成 "安装" 而不是独立于此过程。不支持从网格化环境切换回非网格化环境。

  要使用Isio服务网格、您需要执行以下操作：

  - 添加 `istio-injection:enabled` label 在部署Asta Control Center之前将Asta命名空间添加到Asta命名空间。
  - 使用 `Generic` 入口设置 并为提供备用入口 外部负载平衡。
  - 对于Red Hat OpenShift集群、您需要进行定义 `NetworkAttachmentDefinition` 在所有关联的Astra Control Center名空间上 (`netapp-acc-operator`， `netapp-acc`， `netapp-monitoring` 或任何已替换的自定义卷)。

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

步骤

要安装 Astra 控制中心，请执行以下步骤：

- 下载并提取Astra控制中心
- [如果使用本地注册表、请完成其他步骤]
- [为具有身份验证要求的注册表设置命名空间和密钥]
- 安装 Astra 控制中心操作员
- 配置 Astra 控制中心
- 完成 Astra 控制中心和操作员安装
- [验证系统状态]
- [设置传入以进行负载平衡]
- 登录到 Astra 控制中心 UI

> 请勿删除Astra Control Center运算符(例如、 `kubectl delete -f astra_control_center_operator_deploy.yaml`)、以避免删除Pod。

下载并提取**Astra**控制中心

从以下位置之一下载Astra Control Center映像：

- **Astra**控制服务映像注册表：如果您不对Astra控制中心映像使用本地注册表，或者如果您更喜欢使用此方法

从NetApp 支持站点 下载捆绑包，请使用此选项。

- **Astra**：如果将本地注册表与NetApp 支持站点 控制中心映像一起使用，请使用此选项。

Astra Control图像注册表

1. 登录Asta Control Service。
2. 在信息板上，选择*Deploy a self-managed instance* of Asta Control*。
3. 按照说明登录到Astra Control映像注册表、提取Astra Control Center安装映像并提取该映像。

NetApp 支持站点

1. 下载包含Astra Control Center的软件包 (`astra-control-center-[version].tar.gz`) "Astra Control Center下载页面"。
2. (建议但可选)下载Astra控制中心的证书和签名包 (`astra-control-center-certs-[version].tar.gz`)以验证分发包的签名。

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

此时将显示输出 `Verified OK` 验证成功后。

3. 从Astra Control Center捆绑包中提取映像：

```
tar -vxzf astra-control-center-[version].tar.gz
```

如果使用本地注册表、请完成其他步骤

如果您计划将Astra控制中心捆绑包推送到本地注册表、则需要使用NetApp Astra kubect命令行插件。

安装**NetApp Astra kubectl**插件

要安装最新的NetApp Astra kubecl命令行插件、请完成以下步骤。

开始之前

NetApp可为不同的CPU架构和操作系统提供插件二进制文件。在执行此任务之前、您需要了解您的CPU和操作系统。

如果您已从先前安装中安装了插件、 "确保您已安装最新版本" 在完成这些步骤之前。

步骤

1. 列出可用的NetApp Astra kubect插件二进制文件：

   kubectl插件库是tar包的一部分、并会解压缩到文件夹中 `kubectl-astra`。

   ```
   ls kubectl-astra/
   ```

2. 将操作系统和CPU架构所需的文件移至当前路径、并将其重命名为 `kubectl-astra`：

   ```
   cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
   ```

将映像添加到注册表

1. 如果您计划将Astra Control Center捆绑包推送到本地注册表、请为容器引擎完成相应的步骤顺序：

Docker

a. 更改为tarball的根目录。您应看到 `acc.manifest.bundle.yaml` 文件和以下目录：

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 将Astra Control Center映像目录中的软件包映像推送到本地注册表。在运行之前、请进行以下替换 `push-images` 命令：

- 将<BUNDLE_FILE> 替换为Astra Control捆绑包文件的名称 (`acc.manifest.bundle.yaml`) 。
- 将<MY_FULL_REGISTRY_PATH> 替换为Docker存储库的URL；例如 "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>"。
- 将<MY_REGISTRY_USER> 替换为用户名。
- 将<MY_REGISTRY_TOKEN> 替换为注册表的授权令牌。

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

Podman

a. 更改为tarball的根目录。您应看到此文件和目录：

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 登录到注册表：

```
podman login <YOUR_REGISTRY>
```

c. 准备并运行以下针对您使用的Podman版本自定义的脚本之一。将<MY_FULL_REGISTRY_PATH> 替换为包含任何子目录的存储库的URL。

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

根据您的注册表配置、此脚本创建的映像路径应类似于以下内容：

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 更改目录：

```
cd manifests
```

为具有身份验证要求的注册表设置命名空间和密钥

1. 导出Astra Control Center主机集群的kubeconfig:

```
export KUBECONFIG=[file path]
```

在完成安装之前、请确保您的kubeconfig"指向要安装Astra Control Center的集群。

2. 如果您使用的注册表需要身份验证，则需要执行以下操作：

a. 创建 `NetApp-Acc-operator` 命名空间：

```
kubectl create ns netapp-acc-operator
```

b. 为 `NetApp-Acc-operator` 命名空间创建一个密钥。添加 Docker 信息并运行以下命令：

占位符 `your_registry_path` 应与您先前上传的映像的位置匹配(例如、`[Registry_URL]/netapp/astra/astracc/24.02.0-69`）。

```
kubectl create secret docker-registry astra-registry-cred -n netapp-acc-
operator --docker-server=cr.astra.netapp.io --docker
-username=[astra_account_id] --docker-password=[astra_api_token]
```

+

```
kubectl create secret docker-registry astra-registry-cred -n netapp-acc-
operator --docker-server=[your_registry_path] --docker
-username=[username] --docker-password=[token]
```

+

如果在生成密钥后删除命名空间、请重新创建命名空间、然后重新生成命名空间的密钥。

a. 创建 `netapp-acc` (或自定义命名的)命名空间。

```
kubectl create ns [netapp-acc or custom namespace]
```

b. 为创建密钥 `netapp-acc` (或自定义命名的)命名空间。根据您的注册表首选项、添加Docker信息并运行相应的命令之一：

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=cr.astra.netapp.io --docker-username=[astra_account_id] --docker-password=[astra_api_token]
```

```
kubectl create secret docker-registry astra-registry-cred -n [netapp-acc or custom namespace] --docker-server=[your_registry_path] --docker-username=[username] --docker-password=[token]
```

安装 **Astra** 控制中心操作员

1. (仅限本地注册表)如果使用的是本地注册表、请完成以下步骤：

   a. 打开Asta控制中心操作员部署YAML：

   ```
   vim astra_control_center_operator_deploy.yaml
   ```

   以下步骤将提供一个标注的YAML示例。

   b. 如果您使用的注册表需要身份验证，请将默认行 `imagePullSecs ： []` 替换为以下内容：

   ```
   imagePullSecrets: [{name: astra-registry-cred}]
   ```

   c. 更改 `ASTRA_IMAGE_REGISTRY` 。 `kube-rbac-proxy` 将映像推送到注册表路径中 上一步。

   d. 更改 `ASTRA_IMAGE_REGISTRY` 。 `acc-operator-controller-manager` 将映像推送到注册表路径中 上一步。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  strategy:
    type: Recreate
  template:
```

```
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: ASTRA_IMAGE_REGISTRY/kube-rbac-proxy:v4.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        - name: ACCOP_HELM_INSTALLTIMEOUT
          value: 5m
        image: ASTRA_IMAGE_REGISTRY/acc-operator:24.02.68
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
            memory: 75Mi
```

```
        securityContext:
          allowPrivilegeEscalation: false
      imagePullSecrets: []
      securityContext:
        runAsUser: 65532
      terminationGracePeriodSeconds: 10
```

2. 安装 Astra 控制中心操作员：

```
kubectl apply -f astra_control_center_operator_deploy.yaml
```

展开样本响应：

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.netapp.io created
role.rbac.authorization.k8s.io/acc-operator-leader-election-role
created
clusterrole.rbac.authorization.k8s.io/acc-operator-manager-role
created
clusterrole.rbac.authorization.k8s.io/acc-operator-metrics-reader
created
clusterrole.rbac.authorization.k8s.io/acc-operator-proxy-role
created
rolebinding.rbac.authorization.k8s.io/acc-operator-leader-election-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-manager-
rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/acc-operator-proxy-
rolebinding created
configmap/acc-operator-manager-config created
service/acc-operator-controller-manager-metrics-service created
deployment.apps/acc-operator-controller-manager created
```

3. 验证Pod是否正在运行：

```
kubectl get pods -n netapp-acc-operator
```

配置 **Astra** 控制中心

1. 编辑Astra Control Center自定义资源(CR)文件 (`astra_control_center.yaml`)进行帐户、支持、注册表和其他必要配置：

```
vim astra_control_center.yaml
```

以下步骤将提供一个标注的YAML示例。

2. 修改或确认以下设置：

帐户名称

正在设置 ...	指导	Type	示例
`accountName`	更改 `accountName` 字符串、表示要与Astra Control Center帐户关联的名称。只能有一个accountName。	string	`Example`

AstraVersion

正在设置 ...	指导	Type	示例
`astraVersion`	要部署的Astra控制中心版本。无需对此设置执行任何操作、因为此值将预先填充。	string	`24.02.0-69`

AstraAddress

正在设置 ...	指导	Type	示例
`astraAddress`	更改 `astraAddress` 指向要在浏览器中访问Astra控制中心的FQDN (建议)或IP地址的字符串。此地址用于定义如何在数据中心中找到Astra控制中心、并且与您在完成后从负载平衡器配置的FQDN或IP地址相同 "Astra 控制中心要求"。 注意：请勿使用 `http://` 或 `https://` 地址中。复制此 FQDN 以在中使用 后续步骤。	string	`astra.example.com`

AutoSupport

您在本节中所做的选择将决定您是否参与NetApp的主动支持应用程序Digital Advisor以及数据的发送位置。需要互联网连接(端口442)、所有支持数据均会匿名化。

正在设置 ...	使用 ...	指导	Type	示例
`autoSupport.enrolled`	两者之一 `enrolled` 或 `url` 必须选择字段	更改 `enrolled` 用于将AutoSupport连接到 `false` 对于不具有Internet连接或保留的站点 `true` 对于已连接站点。的设置 `true` 允许将匿名数据发送给NetApp以供支持。默认选择为 `false` 和表示不会向NetApp发送任何支持数据。	布尔值	`false` (此值为默认值)
`autoSupport.url`	两者之一 `enrolled` 或 `url` 必须选择字段	此URL用于确定匿名数据的发送位置。	string	`https://support.netapp.com/asupprod/post/1.0/postAsup`

email

正在设置 ...	指导	Type	示例
`email`	更改 `email` 字符串到默认的初始管理员地址。复制此电子邮件地址以在中使用 后续步骤。此电子邮件地址将用作初始帐户的用户名、用于登录到UI、并在Astra Control中收到事件通知。	string	`admin@example.com`

firstName

正在设置 ...	指导	Type	示例
`firstName`	与Astra帐户关联的默认初始管理员的名字。首次登录后、此处使用的名称将显示在用户界面的标题中。	string	`SRE`

姓氏

正在设置 ...	指导	Type	示例
`lastName`	与Astra帐户关联的默认初始管理员的姓氏。首次登录后、此处使用的名称将显示在用户界面的标题中。	string	`Admin`

imageRegistry

您在本节中的选择定义了托管Astra应用程序映像、Astra控制中心操作员和Astra控制中心Helm存储库的容器映像注册表。

正在设置 ...	使用 ...	指导	Type	示例
`imageRegistry.name`	Required	托管部署Asta Control Center所需的所有映像的Asta Control映像注册表的名称。该值将预先填充、除非配置了本地注册表、否则不需要执行任何操作。对于本地注册表、请将此现有值替换为您在中推送图像的映像注册表的名称 上一步。请勿使用 `http://` 或 `https://` 注册表名称。	string	`cr.astra.netapp.io` (默认) `example.registry.com/astra` (本地注册表示例)
`imageRegistry.secret`	可选	用于通过映像注册表进行身份验证的Kubernetes密钥的名称。该值将预先填充、除非您配置了本地注册表以及在中为该注册表输入的字符串、否则不需要执行任何操作 `imageRegistry.name` 需要密钥。 重要说明：如果您使用的本地注册表不需要授权、则必须将其删除 `secret` 行内 `imageRegistry` 否则安装将失败。	string	`astra-registry-cred`

存储类

正在设置 ...	指导	Type	示例
`storageClass`	更改 `storageClass` 价值来自 `ontap-gold` 安装所需的另一个存储类资源。运行命令 `kubectl get sc` 以确定已配置的现有存储类。必须在清单文件中输入Asta Control配置程序配置的存储类之一 (`astra-control-center-<version>.manifest`)、并将用于Astra PV。如果未设置、则会使用默认存储类。 注意：如果配置了默认存储类、请确保它是唯一具有默认标注的存储类。	string	`ontap-gold`

volumeReclaimPolicy

正在设置 ...	指导	Type	选项
`volumeReclaimPolicy`	这将为Astra的PV设置回收策略。将此策略设置为 `Retain` 删除Astra后保留永久性卷。将此策略设置为 `Delete` 删除Astra后删除永久性卷。如果未设置此值、则会保留PV。	string	• `Retain` (这是默认值) • `Delete`

正在载入类型

正在设置 ...	指导	Type	选项
`ingressType`	请使用以下入口类型之一： 通用 (`ingressType: "Generic"`)(默认) 如果您正在使用另一个入口控制器或希望使用您自己的入口控制器、请使用此选项。部署Astra Control Center后、您需要配置 "入口控制器" 以使用URL公开Astra控制中心。 重要信息：如果您要将服务网格与Astra Control Center结合使用、则必须选择 `Generic` 作为入口类型并设置您自己的 "入口控制器"。 * AccTraefik* (`ingressType: "AccTraefik"`) 如果您不希望配置入口控制器、请使用此选项。这将部署Astra控制中心 `traefik` 网关作为Kubernetes loadbalancer类型的服务。 Astra控制中心使用类型为"loadbalancer"的服务 (`svc/traefik`)、并要求为其分配可访问的外部IP地址。如果您的环境允许使用负载平衡器、但您尚未配置一个平衡器、则可以使用MetalLB或其他外部服务负载平衡器为该服务分配外部IP地址。在内部 DNS 服务器配置中，您应将为 Astra 控制中心选择的 DNS 名称指向负载平衡的 IP 地址。 注意：有关"load平衡器"和传入服务类型的详细信息、请参见 "要求"。	string	• `Generic` (这是默认值) • `AccTraefik`

可扩展大小

正在设置 ...	指导	Type	选项
`scaleSize`	默认情况下、Astra将使用高可用性(HA) `scaleSize` 的 `Medium` ，可在HA中部署大多数服务，并部署多个副本以实现冗余。使用 `scaleSize` 作为 `Small`A作用 是减少所有服务的副本数量，但主要服务除外，以减少使用量。提示： `Medium` 部署包含大约100个Pod (不包括瞬时工作负载)。100个Pod基于一个三主节点和三个工作节点配置)。请注意您问题描述 的环境中可能存在的每POD网络限制限制、尤其是在考虑灾难恢复方案时。	string	• `Small` • `Medium` (这是默认值)

AstraResourcesScal

正在设置 ...	指导	Type	选项
`astraResourcesScaler`	AstraControlCenter资源限制的扩展选项。默认情况下、Astra控制中心会进行部署、并为Astra中的大多数组件设置了资源请求。通过这种配置、Astra控制中心软件堆栈可以在应用程序负载和扩展性增加的环境中更好地运行。但是、在使用较小的开发或测试集群的情况下、CR字段为 `astraResourcesScalar` 可设置为 `Off`。此操作将禁用资源请求、并允许在较小的集群上部署。	string	• `Default` (这是默认值) • `Off`

其他值

将以下附加值添加到Astra控制中心CR中、以防止安装已知问题描述：

```
additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

CRD

您在本节中的选择决定了Astra控制中心应如何处理CRD。

正在设置 ...	指导	**Type**	示例
`crds.externalCertManager`	如果使用外部证书管理器、请进行更改 `externalCertManager` to `true`。默认值 `false` 使Astra控制中心在安装期间安装自己的证书管理器CRD。CRD是集群范围的对象、安装它们可能会影响集群的其他部分。您可以使用此标志向Astra控制中心发出信号、指示这些CRD将由Astra控制中心以外的集群管理员安装和管理。	布尔值	`False` (此值为默认值)
`crds.externalTraefik`	默认情况下、Astra控制中心将安装所需的Traefik CRD。CRD是集群范围的对象、安装它们可能会影响集群的其他部分。您可以使用此标志向Astra控制中心发出信号、指示这些CRD将由Astra控制中心以外的集群管理员安装和管理。	布尔值	`False` (此值为默认值)

在完成安装之前、请确保为您的配置选择了正确的存储类和入口类型。

示例**Astra_control_cCenter.yaml**

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[cr.astra.netapp.io or your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
  volumeReclaimPolicy: "Retain"
  ingressType: "Generic"
  scaleSize: "Medium"
  astraResourcesScaler: "Default"
  additionalValues:
    keycloak-operator:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  crds:
    externalTraefik: false
    externalCertManager: false
```

完成 **Astra** 控制中心和操作员安装

1. 如果您在上一步中尚未创建，请创建 `NetApp-Accc` （或自定义）命名空间：

```
kubectl create ns [netapp-acc or custom namespace]
```

2. 如果您正在Astra Control Center中使用服务网格、请将以下标签添加到 `netapp-acc` 或自定义命名空间：

> 您的入口类型 (`ingressType`)必须设置为 `Generic` 在Astra Control Center CR中、然后继续执行此命令。

```
kubectl label ns [netapp-acc or custom namespace] istio-
injection:enabled
```

3. （建议） "启用严格的MTLS" 对于Isio service Mesh：

```
kubectl apply -n istio-system -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
spec:
  mtls:
    mode: STRICT
EOF
```

4. 在 `NetApp-Accc` （或您的自定义）命名空间中安装 Astra Control Center：

```
kubectl apply -f astra_control_center.yaml -n [netapp-acc or custom
namespace]
```

A作用 力控制中心操作员将自动检查环境要求。缺少 "要求" 发生原因 您的安装是否失败或Astra控制中心是否无法正常运行。请参见 下一节 检查与自动系统检查相关的警告消息。

验证系统状态

您可以使用kubectl命令验证系统状态。如果您更喜欢使用 OpenShift ，则可以使用同等的 oc 命令执行验证步骤。

步骤

1. 验证安装过程是否未生成与验证检查相关的警告消息：

```
kubectl get acc [astra or custom Astra Control Center CR name] -n
[netapp-acc or custom namespace] -o yaml
```

A作用 力控制中心操作员日志中还会报告其他警告消息。

2. 更正自动需求检查报告的环境中的任何问题。

您可以通过确保环境满足来更正问题 "要求" A作用 控制中心。

3. 验证是否已成功安装所有系统组件。

```
kubectl get pods -n [netapp-acc or custom namespace]
```

每个 POD 的状态应为 `running` 。部署系统 Pod 可能需要几分钟的时间。

展开以显示样本响应

```
acc-helm-repo-5bd77c9ddd-8wxm2              1/1     Running     0
1h
activity-5bb474dc67-8l9ss                   1/1     Running     0
1h
activity-5bb474dc67-qbrtq                   1/1     Running     0
1h
api-token-authentication-6wbj2              1/1     Running     0
1h
api-token-authentication-9pgw6              1/1     Running     0
1h
api-token-authentication-tqf6d              1/1     Running     0
1h
asup-5495f44dbd-z4kft                       1/1     Running     0
1h
authentication-6fdd899858-5x45s             1/1     Running     0
1h
bucketservice-84d47487d-n9xgp               1/1     Running     0
1h
bucketservice-84d47487d-t5jhm               1/1     Running     0
1h
cert-manager-5dcb7648c4-hbldc               1/1     Running     0
1h
cert-manager-5dcb7648c4-nr9qf               1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-bk2tf    1/1     Running     0
1h
cert-manager-cainjector-59b666fb75-pfnck    1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-ngz2x        1/1     Running     0
1h
cert-manager-webhook-c6f9b6796-rwtbn        1/1     Running     0
1h
certificates-5f5b7b4dd-52tnj                1/1     Running     0
1h
certificates-5f5b7b4dd-gtjbx                1/1     Running     0
1h
certificates-expiry-check-28477260-dz5vw    0/1     Completed   0
1h
cloud-extension-6f58cc579c-lzfmv            1/1     Running     0
1h
cloud-extension-6f58cc579c-zw2km            1/1     Running     0
1h
cluster-orchestrator-79dd5c8d95-qjg92       1/1     Running     0
```

```
1h
composite-compute-85dc84579c-nz82f          1/1     Running     0
1h
composite-compute-85dc84579c-wx2z2          1/1     Running     0
1h
composite-volume-bff6f4f76-789nj            1/1     Running     0
1h
composite-volume-bff6f4f76-kwnd4            1/1     Running     0
1h
credentials-79fd64f788-m7m8f                1/1     Running     0
1h
credentials-79fd64f788-qnc6c                1/1     Running     0
1h
entitlement-f69cdbd77-4p2kn                 1/1     Running     0
1h
entitlement-f69cdbd77-hswm6                 1/1     Running     0
1h
features-7b9585444c-7xd7m                   1/1     Running     0
1h
features-7b9585444c-dcqwc                   1/1     Running     0
1h
fluent-bit-ds-crq8m                         1/1     Running     0
1h
fluent-bit-ds-gmgq8                         1/1     Running     0
1h
fluent-bit-ds-gzr4f                         1/1     Running     0
1h
fluent-bit-ds-j6sf6                         1/1     Running     0
1h
fluent-bit-ds-v4t9f                         1/1     Running     0
1h
fluent-bit-ds-x7j59                         1/1     Running     0
1h
graphql-server-6cc684fb46-2x8lr             1/1     Running     0
1h
graphql-server-6cc684fb46-bshbd             1/1     Running     0
1h
hybridauth-84599f79fd-fjc7k                 1/1     Running     0
1h
hybridauth-84599f79fd-s9pmn                 1/1     Running     0
1h
identity-95df98cb5-dvlmz                    1/1     Running     0
1h
identity-95df98cb5-krf59                    1/1     Running     0
1h
influxdb2-0                                 1/1     Running     0
```

```
1h
keycloak-operator-6d4d688697-cfq8b          1/1     Running     0
1h
krakend-5d5c8f4668-7bq8g                    1/1     Running     0
1h
krakend-5d5c8f4668-t8hbn                    1/1     Running     0
1h
license-689cdd4595-2gsc8                    1/1     Running     0
1h
license-689cdd4595-g6vwk                    1/1     Running     0
1h
login-ui-57bb599956-4fwgz                   1/1     Running     0
1h
login-ui-57bb599956-rhztb                   1/1     Running     0
1h
loki-0                                      1/1     Running     0
1h
metrics-facade-846999bdd4-f7jdm             1/1     Running     0
1h
metrics-facade-846999bdd4-lnsxl             1/1     Running     0
1h
monitoring-operator-6c9d6c4b8c-ggkrl        2/2     Running     0
1h
nats-0                                      1/1     Running     0
1h
nats-1                                      1/1     Running     0
1h
nats-2                                      1/1     Running     0
1h
natssync-server-6df7d6cc68-9v2gd            1/1     Running     0
1h
nautilus-64b7fbdd98-bsgwb                   1/1     Running     0
1h
nautilus-64b7fbdd98-djlhw                   1/1     Running     0
1h
openapi-864584bccc-75nlv                    1/1     Running     0
1h
openapi-864584bccc-zh6bx                    1/1     Running     0
1h
polaris-consul-consul-server-0              1/1     Running     0
1h
polaris-consul-consul-server-1              1/1     Running     0
1h
polaris-consul-consul-server-2              1/1     Running     0
1h
polaris-keycloak-0                          1/1     Running     2 (1h
```

```
ago)      1h
polaris-keycloak-1                            1/1     Running     0
1h
polaris-keycloak-db-0                         1/1     Running     0
1h
polaris-keycloak-db-1                         1/1     Running     0
1h
polaris-keycloak-db-2                         1/1     Running     0
1h
polaris-mongodb-0                             1/1     Running     0
1h
polaris-mongodb-1                             1/1     Running     0
1h
polaris-mongodb-2                             1/1     Running     0
1h
polaris-ui-66476dcf87-f6s8j                   1/1     Running     0
1h
polaris-ui-66476dcf87-ztjk7                   1/1     Running     0
1h
polaris-vault-0                               1/1     Running     0
1h
polaris-vault-1                               1/1     Running     0
1h
polaris-vault-2                               1/1     Running     0
1h
public-metrics-bfc4fc964-x4m79                1/1     Running     0
1h
storage-backend-metrics-7dbb88d4bc-g78cj      1/1     Running     0
1h
storage-provider-5969b5df5-hjvcm              1/1     Running     0
1h
storage-provider-5969b5df5-r79ld              1/1     Running     0
1h
task-service-5fc9dc8d99-4q4f4                 1/1     Running     0
1h
task-service-5fc9dc8d99-8l5zl                 1/1     Running     0
1h
task-service-task-purge-28485735-fdzkd        1/1     Running     0
12m
telegraf-ds-2rgm4                             1/1     Running     0
1h
telegraf-ds-4qp6r                             1/1     Running     0
1h
telegraf-ds-77frs                             1/1     Running     0
1h
telegraf-ds-bc725                             1/1     Running     0
```

```
1h
telegraf-ds-cvmxf                              1/1     Running     0
1h
telegraf-ds-tqzgj                              1/1     Running     0
1h
telegraf-rs-5wtd8                              1/1     Running     0
1h
telemetry-service-6747866474-5djnc             1/1     Running     0
1h
telemetry-service-6747866474-thb7r             1/1     Running     1 (1h
ago)      1h
tenancy-5669854fb6-gzdzf                       1/1     Running     0
1h
tenancy-5669854fb6-xvsm2                       1/1     Running     0
1h
traefik-8f55f7d5d-4lgfw                        1/1     Running     0
1h
traefik-8f55f7d5d-j4wt6                        1/1     Running     0
1h
traefik-8f55f7d5d-p6gcq                        1/1     Running     0
1h
trident-svc-7cb5bb4685-54cnq                   1/1     Running     0
1h
trident-svc-7cb5bb4685-b28xh                   1/1     Running     0
1h
vault-controller-777b9bbf88-b5bqt              1/1     Running     0
1h
vault-controller-777b9bbf88-fdfd8              1/1     Running     0
1h
```

4. (可选)观看 `acc-operator` 用于监控进度的日志：

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

> `accHost` 集群注册是最后一项操作、如果失败、发生原因 部署不会失败。如果日志中指示的集群注册失败、您可以尝试通过重新注册 "在UI中添加集群工作流" 或 API 。

5. 在所有Pod运行时、验证安装是否成功 (`READY` 为 `True`)并获取登录Astra Control Center时要使用的初始设置密码：

```
kubectl get AstraControlCenter -n [netapp-acc or custom namespace]
```

响应：

```
NAME    UUID                                   VERSION     ADDRESS
READY
astra   9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  24.02.0-69
10.111.111.111  True
```

复制UUID值。密码为 `Acc-` ，后跟 UUID 值（`Acc-UUID` 或在此示例中为 `Acc-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f` ）。

设置传入以进行负载平衡

您可以设置一个Kubernetes入口控制器、用于管理对服务的外部访问。如果您使用的是默认值、则以下过程提供了入口控制器的设置示例 `ingressType: "Generic"` 在Astra Control Center自定义资源中 (`astra_control_center.yaml`）。如果指定、则不需要使用此操作步骤 `ingressType: "AccTraefik"` 在Astra Control Center自定义资源中 (`astra_control_center.yaml`）。

部署Astra Control Center后、您需要配置传入控制器、以便使用URL公开Astra Control Center。

设置步骤因所使用的入口控制器类型而异。Astra控制中心支持多种传入控制器类型。这些设置过程提供了一些常见传入控制器类型的示例步骤。

开始之前

- 所需 "入口控制器" 应已部署。
- 。 "入口类" 应已创建与入口控制器对应的。

Istio入口的步骤

1. 配置Istio入口。

   此操作步骤 假定使用"默认"配置文件部署Istio。

2. 为传入网关收集或创建所需的证书和专用密钥文件。

   您可以使用CA签名或自签名证书。公用名必须为Astra地址(FQDN)。

   命令示例：

   ```
   openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
   tls.crt
   ```

3. 创建密钥 `tls secret name` 类型 `kubernetes.io/tls` 中的TLS专用密钥和证书 `istio-system namespace` 如TLS机密中所述。

   命令示例：

```
kubectl create secret tls [tls secret name] --key="tls.key"
--cert="tls.crt" -n istio-system
```

密钥名称应与`istio-Infile.yaml`文件中提供的`spec.tls.secretName`匹配。

4. 在中部署入站资源 `netapp-acc` (或自定义命名的)命名空间 (`istio-Ingress.yaml` 在此示例中使用)：

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. 应用更改：

```
kubectl apply -f istio-Ingress.yaml
```

6. 检查入口状态：

```
kubectl get ingress -n [netapp-acc or custom namespace]
```

响应：

```
NAME    CLASS HOSTS              ADDRESS         PORTS   AGE
ingress istio astra.example.com 172.16.103.248  80, 443 1h
```

7. 完成Astra控制中心安装。

nginx 入口控制器的步骤

1. 创建类型的密钥 `kubernetes.io/tls` 中的TLS专用密钥和证书 `netapp-acc` (或自定义命名的)命名空间、如中所述 "TLS 密钥"。
2. 在中部署传入资源 `netapp-acc` (或自定义命名的)命名空间 (`nginx-Ingress.yaml` 在此示例中使用)：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: netapp-acc-ingress
  namespace: [netapp-acc or custom namespace]
spec:
  ingressClassName: [class name for nginx controller]
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: <ACC address>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

3. 应用更改：

```
kubectl apply -f nginx-Ingress.yaml
```

NetApp建议将nginx控制器安装为部署、而不是安装 `daemonSet`。

OpenShift 入口控制器的步骤

1. 获取证书并获取密钥，证书和 CA 文件，以供 OpenShift 路由使用。
2. 创建 OpenShift 路由：

```
oc create route edge --service=traefik --port=web -n [netapp-acc or
custom namespace] --insecure-policy=Redirect --hostname=<ACC address>
--cert=cert.pem --key=key.pem
```

登录到 **Astra** 控制中心 **UI**

安装Astra Control Center后、您将更改默认管理员的密码、并登录到Astra Control Center UI信息板。

步骤

1. 在浏览器中、输入FQDN (包括 `https://` 前缀) `astraAddress` 在中 `astra_control_center.yaml` CR时间 您安装了 Astra 控制中心。
2. 如果出现提示、请接受自签名证书。

   您可以在登录后创建自定义证书。

3. 在Astra Control Center登录页面上、输入您用于的值 `email` 在中 `astra_control_center.yaml` CR时间 您安装了 Astra 控制中心、后跟初始设置密码 (`ACC-[UUID]`）。

   如果您输入的密码三次不正确，管理员帐户将锁定 15 分钟。

4. 选择 * 登录 * 。
5. 根据提示更改密码。

   如果这是您第一次登录、但您忘记了密码、并且尚未创建任何其他管理用户帐户、请联系 "NetApp 支持" 以获得密码恢复帮助。

6. （可选）删除现有自签名 TLS 证书并将其替换为 "由证书颁发机构（ CA ）签名的自定义 TLS 证书"。

对安装进行故障排除

如果任何服务处于 `Error` 状态，您可以检查日志。查找 400 到 500 范围内的 API 响应代码。这些信息表示发生故障的位置。

选项

- 要检查 Astra 控制中心操作员日志，请输入以下内容：

```
kubectl logs deploy/acc-operator-controller-manager -n netapp-acc-
operator -c manager -f
```

- 要检查Asta Control Center CR的输出：

```
kubectl get acc -n [netapp-acc or custom namespace] -o yaml
```

其他安装过程

- 使用**Red Hat OpenShift OperatorHub**进行安装：使用此方法 "备用操作步骤" 使用OperatorHub在OpenShift上安装A作用 力控制中心。
- 使用**Cloud Volumes ONTAP** 后端在公有 云中安装：使用 "这些过程" 在带有Cloud Volumes ONTAP 存储后端的Amazon Web Services (AWS)、Google云平台(GCP)或Microsoft Azure中安装Astra控制中心。

下一步行动

- (可选)根据您的环境、完成安装后操作 "配置步骤"。
- "安装Astra Control Center、登录到UI并更改密码后、您需要设置许可证、添加集群、启用身份验证、管理存储以及添加存储分段"。

配置外部证书管理器

如果Kubernetes集群中已存在证书管理器、则需要执行一些前提步骤、以使Astra控制中心不会安装自己的证书管理器。

步骤

1. 确认已安装证书管理器：

```
kubectl get pods -A | grep 'cert-manager'
```

响应示例：

```
cert-manager   essential-cert-manager-84446f49d5-sf2zd              1/1
Running    0      6d5h
cert-manager   essential-cert-manager-cainjector-66dc99cc56-9ldmt   1/1
Running    0      6d5h
cert-manager   essential-cert-manager-webhook-56b76db9cc-fjqrq      1/1
Running    0      6d5h
```

2. 为创建证书/密钥对 `astraAddress` FQDN：

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout tls.key -out
tls.crt
```

响应示例：

```
Generating a 2048 bit RSA private key
..................+++
.........................................+++
writing new private key to 'tls.key'
```

3. 使用先前生成的文件创建密钥：

```
kubectl create secret tls selfsigned-tls --key tls.key --cert tls.crt -n
<cert-manager-namespace>
```

响应示例：

```
secret/selfsigned-tls created
```

4. 创建 `ClusterIssuer` 文件*精确*如下、但包含的命名空间位置 `cert-manager` Pod的安装：

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: astra-ca-clusterissuer
  namespace: <cert-manager-namespace>
spec:
  ca:
    secretName: selfsigned-tls
```

```
kubectl apply -f ClusterIssuer.yaml
```

响应示例：

```
clusterissuer.cert-manager.io/astra-ca-clusterissuer created
```

5. 验证是否已 `ClusterIssuer` 已正确启动。 `Ready` 必须为 `True` 在继续操作之前：

```
kubectl get ClusterIssuer
```

响应示例：

```
NAME                        READY   AGE
astra-ca-clusterissuer      True    9s
```

6. 完成 "Astra 控制中心安装过程"。有一个 "Astra控制中心集群YAML的所需配置步骤" 其中、您可以更改CRD值以指示证书管理器是外部安装的。您必须在安装期间完成此步骤、以使Astra控制中心能够识别外部证书管理器。

使用 OpenShift OperatorHub 安装 Astra 控制中心

如果您使用的是 Red Hat OpenShift ，则可以使用 Red Hat 认证操作员安装 Astra Control Center 。使用此操作步骤从安装 Astra 控制中心 "Red Hat 生态系统目录" 或使用 Red Hat OpenShift 容器平台。

完成此操作步骤后，您必须返回到安装操作步骤以完成 "剩余步骤" 以验证安装是否成功并登录。

开始之前

- 满足环境前提条件： "开始安装之前，请为 Astra Control Center 部署准备您的环境"。

> 在第三个容错域或二级站点中部署A作用 力控制中心。对于应用程序复制和无缝灾难恢复、建议执行此操作。

- 确保集群操作员和**API**服务运行正常：
  - 在OpenShift集群中、确保所有集群操作员均处于运行状况良好的状态：

```
oc get clusteroperators
```

  - 在OpenShift集群中、确保所有API服务均处于运行状况良好的状态：

```
oc get apiservices
```

- 确保具有可路由的**FQDN**：您计划使用的Astra FQDN可以路由到集群。这意味着您的内部 DNS 服务器中有一个 DNS 条目，或者您正在使用已注册的核心 URL 路由。
- 获取**OpenShift**权限：要执行所述的安装步骤、您需要拥有对Red Hat OpenShift容器平台的所有必要权限和访问权限。
- 配置证书管理器：如果集群中已存在证书管理器，则需要执行某些操作 "前提条件步骤" 这样、Astra控制中心就不会安装自己的证书管理器。默认情况下、Astra控制中心会在安装期间安装自己的证书管理器。
- 设置**Kubernetes**入口控制器：如果您有一个Kubernetes入口控制器来管理对服务的外部访问、例如集群中的负载平衡、则需要将其设置为与Astra Control Center配合使用：

a. 创建操作员命名空间：

```
oc create namespace netapp-acc-operator
```

b. "完成设置" 适用于您的入口控制器类型。

- **(仅限ONTAP SAN驱动程序)启用多路径：**如果使用的是ONTAP SAN驱动程序、请确保在所有Kubernetes集群上启用了多路径。

您还应考虑以下事项：

- 获取**NetApp Astra**控件映像注册表的访问权限：

  您可以选择从NetApp映像注册表中获取Astra控件的安装映像和增强功能、例如Astra控件配置程序。

  a. 记录您登录注册表所需的Astra Control帐户ID。

     您可以在Astra Control Service Web UI中查看您的帐户ID。选择页面右上角的图图标，选择*API access*并记下您的帐户ID。

  b. 在同一页面中，选择*Generate API t令牌*并将API令牌字符串复制到剪贴板，然后将其保存在编辑器中。

  c. 登录到Asta Control注册表：

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

- 安装用于安全通信的服务网格：强烈建议使用保护Astra Control主机集群通信通道的安全 "支持的服务网格"。

  只能在Astra Control Center期间将Astra Control Center与服务网格集成 "安装" 而不是独立于此过程。不支持从网格化环境切换回非网格化环境。

  要使用Isio服务网格、您需要执行以下操作：

  - 添加 `istio-injection:enabled` 在部署Asta Control Center之前、请标记Asta命名空间。
  - 使用 `Generic` 入口设置 并为提供备用入口 "外部负载平衡"。
  - 对于Red Hat OpenShift集群、您需要进行定义 `NetworkAttachmentDefinition` 在所有关联的Astra Control Center名空间上 (`netapp-acc-operator`， `netapp-acc`， `netapp-monitoring` 或任何已替换的自定义卷)。

```
cat <<EOF | oc -n netapp-acc-operator create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-acc create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF

cat <<EOF | oc -n netapp-monitoring create -f -
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: istio-cni
EOF
```

步骤

- 下载并提取Astra控制中心
- [如果使用本地注册表、请完成其他步骤]
- [找到操作员安装页面]
- [安装操作员]
- 安装 Astra 控制中心

> 请勿删除Astra Control Center运算符(例如、 `kubectl delete -f astra_control_center_operator_deploy.yaml`)、以避免删除Pod。

下载并提取**Astra**控制中心

从以下位置之一下载Astra Control Center映像：

- **Astra**控制服务映像注册表：如果您不对Astra控制中心映像使用本地注册表，或者如果您更喜欢使用此方法从NetApp 支持站点 下载捆绑包，请使用此选项。
- **Astra**：如果将本地注册表与NetApp 支持站点 控制中心映像一起使用，请使用此选项。

Astra Control图像注册表

1. 登录Asta Control Service。
2. 在信息板上，选择*Deploy a self-managed instance* of Asta Control*。
3. 按照说明登录到Astra Control映像注册表、提取Astra Control Center安装映像并提取该映像。

NetApp 支持站点

1. 下载包含Astra Control Center的软件包 (`astra-control-center-[version].tar.gz`) "Astra Control Center下载页面"。
2. (建议但可选)下载Astra控制中心的证书和签名包 (`astra-control-center-certs-[version].tar.gz`)以验证分发包的签名。

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenter-public.pub
-signature certs/astra-control-center-[version].tar.gz.sig astra-
control-center-[version].tar.gz
```

此时将显示输出 `Verified OK` 验证成功后。

3. 从Astra Control Center捆绑包中提取映像：

```
tar -vxzf astra-control-center-[version].tar.gz
```

如果使用本地注册表、请完成其他步骤

如果您计划将Astra控制中心捆绑包推送到本地注册表、则需要使用NetApp Astra kubect命令行插件。

安装**NetApp Astra kubectl**插件

要安装最新的NetApp Astra kubecl命令行插件、请完成以下步骤。

开始之前

NetApp可为不同的CPU架构和操作系统提供插件二进制文件。在执行此任务之前、您需要了解您的CPU和操作系统。

如果您已从先前安装中安装了插件、 "确保您已安装最新版本" 在完成这些步骤之前。

步骤

1. 列出可用的NetApp Astra kubectl插件二进制文件、并记下操作系统和CPU架构所需的文件名称：

kubectl插件库是tar包的一部分、并会解压缩到文件夹中 `kubectl-astra`。

```
ls kubectl-astra/
```

2. 将正确的二进制文件移动到当前路径并重命名为 `kubectl-astra`：

```
cp kubectl-astra/<binary-name> /usr/local/bin/kubectl-astra
```

将映像添加到注册表

1. 如果您计划将Astra Control Center捆绑包推送到本地注册表、请为容器引擎完成相应的步骤顺序：

Docker

a. 更改为tarball的根目录。您应看到 `acc.manifest.bundle.yaml` 文件和以下目录：

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 将Astra Control Center映像目录中的软件包映像推送到本地注册表。在运行之前、请进行以下替换 `push-images` 命令：

- 将<BUNDLE_FILE> 替换为Astra Control捆绑包文件的名称 (`acc.manifest.bundle.yaml`) 。
- 将<MY_FULL_REGISTRY_PATH> 替换为Docker存储库的URL；例如 "<a href="https://<docker-registry>"" class="bare">https://<docker-registry>"。
- 将<MY_REGISTRY_USER> 替换为用户名。
- 将<MY_REGISTRY_TOKEN> 替换为注册表的授权令牌。

```
kubectl astra packages push-images -m <BUNDLE_FILE> -r
<MY_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

Podman

a. 更改为tarball的根目录。您应看到此文件和目录：

```
acc/
kubectl-astra/
acc.manifest.bundle.yaml
```

b. 登录到注册表：

```
podman login <YOUR_REGISTRY>
```

c. 准备并运行以下针对您使用的Podman版本自定义的脚本之一。将<MY_FULL_REGISTRY_PATH> 替换为包含任何子目录的存储库的URL。

```
<strong>Podman 4</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=24.02.0-69
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

根据您的注册表配置、此脚本创建的映像路径应类似于以下内容：

```
https://downloads.example.io/docker-astra-control-
prod/netapp/astra/acc/24.02.0-69/image:version
```

2. 更改目录：

```
cd manifests
```

找到操作员安装页面

1. 要访问操作员安装页面，请完成以下过程之一：

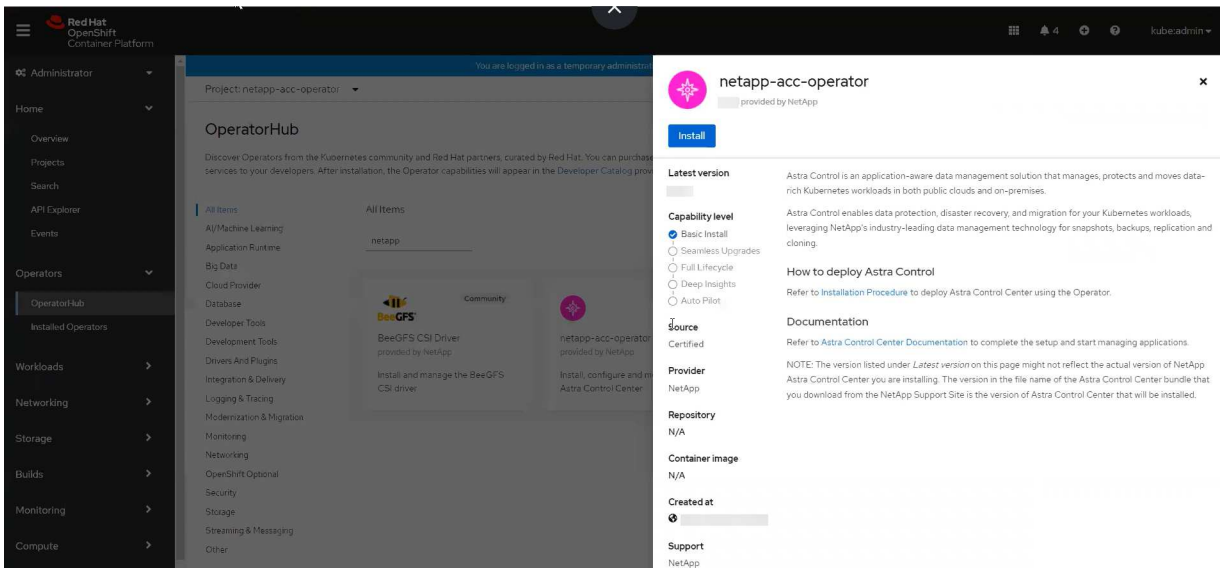
Red Hat OpenShift Web控制台

a. 登录到 OpenShift 容器平台 UI 。

b. 从侧面菜单中，选择 * 运算符 > OperatorHub * 。

> 使用此运算符只能升级到Astra Control Center的当前版本。

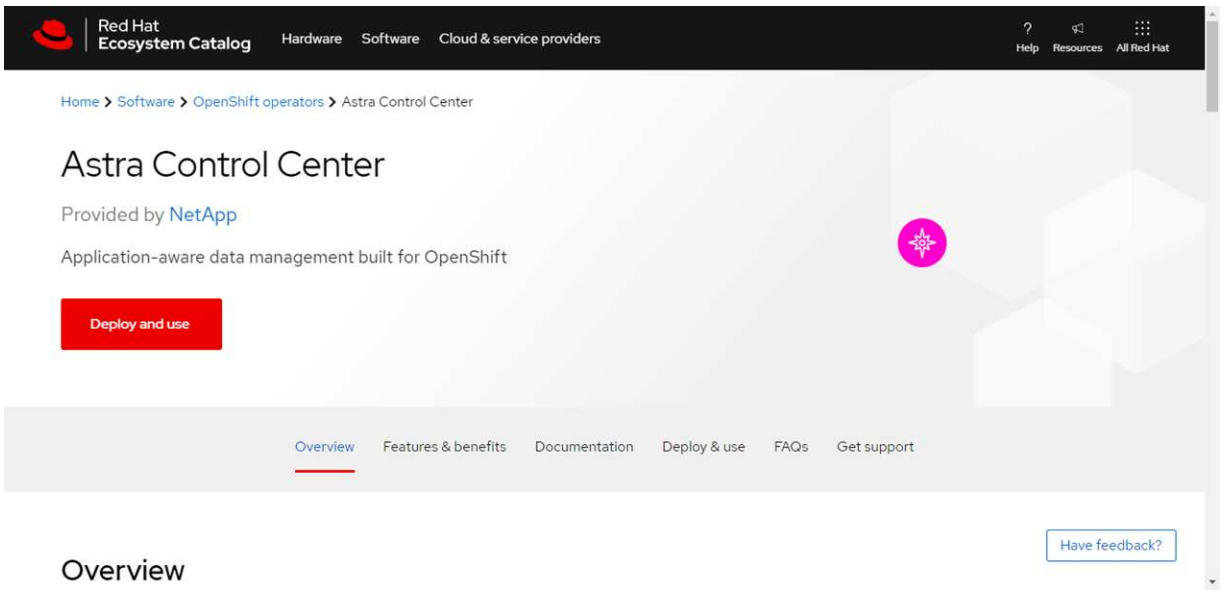
c. 搜索 `netapp-acc` 并选择NetApp Asta控制中心操作员。



Red Hat 生态系统目录

a. 选择 NetApp Astra 控制中心 "运算符"。

b. 选择*部署和使用*。



安装操作员

1. 完成 * 安装操作员 * 页面并安装操作员：

   操作员将在所有集群命名空间中可用。

   a. 选择运算符命名空间或 `netapp-ac-operator` namespace will be created automatically as part of the operator install.

   b. 选择手动或自动批准策略。

      建议手动批准。每个集群只能运行一个操作员实例。

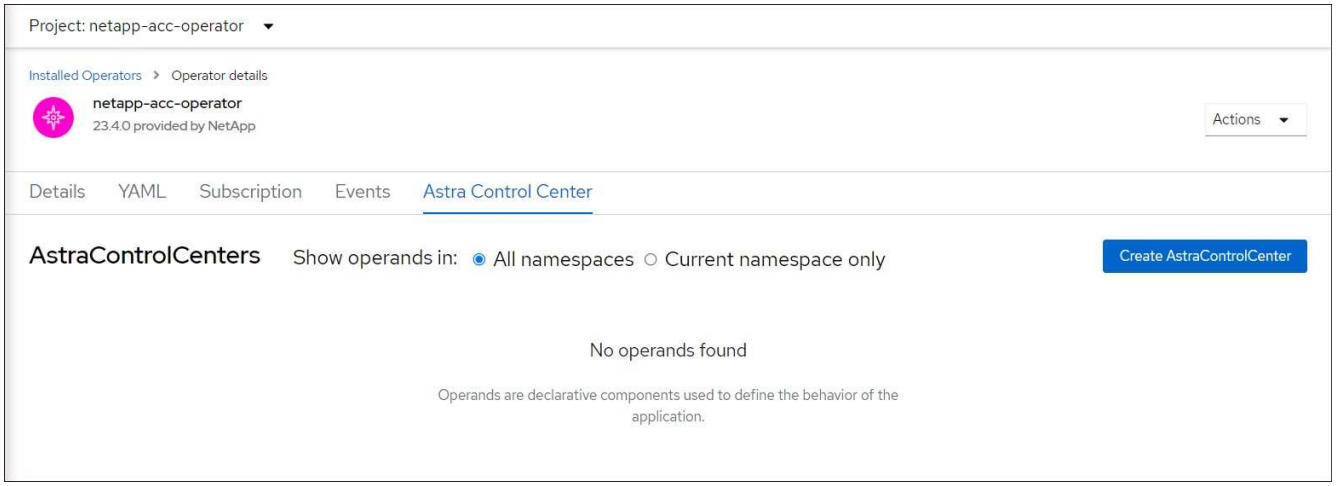
   c. 选择 * 安装 * 。

      如果您选择了手动批准策略、系统将提示您批准此操作员的手动安装计划。

2. 从控制台中，转到 OperatorHub 菜单并确认操作员已成功安装。

安装 **Astra** 控制中心

1. 从Astra Control Center操作员的* Astra Control Center*选项卡中的控制台中、选择*创建AstraControlCenter*。



2. 填写 `Create AstraControlCenter` Form 字段：

   a. 保留或调整 Astra 控制中心名称。

   b. 为Astra控制中心添加标签。

   c. 启用或禁用自动支持。建议保留自动支持功能。

   d. 输入Astra控制中心FQDN或IP地址。请止步 `http://` 或 `https://` 在地址字段中。

   e. 输入Asta Control Center版本、例如24.02.0-69。

   f. 输入帐户名称，电子邮件地址和管理员姓氏。

   g. 选择的卷回收策略 `Retain`， `Recycle`或 `Delete`。默认值为 `Retain`。

   h. 选择安装的比例大小。

默认情况下、Astra将使用高可用性(HA) `scaleSize` 的 `Medium`，可在HA中部署大多数服务，并部署多个副本以实现冗余。使用 `scaleSize` 作为 `Small`A作用 是减少所有服务的副本数量，但主要服务除外，以减少使用量。

i. 选择入口类型：

- 通用 (`ingressType: "Generic"`)(默认)

  如果您正在使用另一个入口控制器或希望使用您自己的入口控制器、请使用此选项。部署Astra Control Center后、您需要配置 "入口控制器" 以使用URL公开Astra控制中心。

- * AccTraefik* (`ingressType: "AccTraefik"`)

  如果您不希望配置入口控制器、请使用此选项。这将部署Astra控制中心 `traefik` 网关作为Kubernetes的"loadbalancer"类型服务。

Astra控制中心使用类型为"loadbalancer"的服务 (`svc/traefik`)、并要求为其分配可访问的外部IP地址。如果您的环境允许使用负载平衡器、但您尚未配置一个平衡器、则可以使用MetalLB或其他外部服务负载平衡器为该服务分配外部IP地址。在内部 DNS 服务器配置中，您应将为 Astra 控制中心选择的 DNS 名称指向负载平衡的 IP 地址。

有关"负载平衡器"和传入服务类型的详细信息、请参见 "要求"。

a. 在*Image Registry*中，除非配置了本地注册表，否则请使用默认值。对于本地注册表、请将此值替换为您在上一步中推送映像的本地映像注册表路径。请止步 `http://` 或 `https://` 在地址字段中。

b. 如果您使用的映像注册表需要身份验证、请输入映像密钥。

如果您使用的注册表需要身份验证、 在集群上创建密钥。

c. 输入管理员的名字。

d. 配置资源扩展。

e. 提供默认存储类。

如果配置了默认存储类、请确保它是唯一具有默认标注的存储类。

f. 定义 CRD 处理首选项。

3. 选择YAML视图以查看您选择的设置。

4. 选择 `Create` 。

创建注册表密钥

如果您使用的注册表需要进行身份验证、请在OpenShift集群上创建一个密钥、然后在中输入该密钥名称 `Create AstraControlCenter` 表单字段。

1. 为Astra控制中心操作员创建命名空间：

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. 在此命名空间中创建密钥：

```
oc create secret docker-registry astra-registry-cred -n [netapp-acc-
operator or custom namespace] --docker-server=[your_registry_path]
--docker username=[username] --docker-password=[token]
```

Astra Control仅支持Docker注册表机密。

3. 完成中的其余字段 创建AstraControlCenter表单字段。

下一步行动

完成 "剩余步骤" 要验证是否已成功安装Astra控制中心、请设置一个入口控制器(可选)并登录到UI。此外、您还需要执行 "设置任务" 完成安装后。

使用 **Cloud Volumes ONTAP** 存储后端安装 **Astra** 控制中心

借助 Astra 控制中心，您可以使用自管理的 Kubernetes 集群和 Cloud Volumes ONTAP 实例在混合云环境中管理应用程序。您可以在内部Kubornetes集群中或云环境中的一个自行管理的Kubornetes集群中部署Astra Control Center。

在其中一种部署中，您可以使用 Cloud Volumes ONTAP 作为存储后端来执行应用程序数据管理操作。您还可以将 S3 存储分段配置为备份目标。

要在Amazon Web Services (AWS)、Google云平台(GCP)和Microsoft Azure中使用Cloud Volumes ONTAP 存储后端安装Astra控制中心、请根据您的云环境执行以下步骤。

- 在 Amazon Web Services 中部署 Astra 控制中心
- 在Google Cloud Platform中部署Astra控制中心
- 在 Microsoft Azure 中部署 Astra 控制中心

您可以使用自管理Kubernetes集群(例如OpenShift容器平台(OCP))在分发版中管理应用程序。只有自管理的OCP集群才会通过验证来部署Astra控制中心。

在 **Amazon Web Services** 中部署 **Astra** 控制中心

您可以在 Amazon Web Services （ AWS ）公有 云上托管的自管理 Kubernetes 集群上部署 Astra 控制中心。

AWS所需的功能

在AWS中部署Astra Control Center之前、您需要满足以下条件：

- Astra Control Center 许可证。请参见 "Astra 控制中心许可要求"。

- "满足 Astra 控制中心的要求"。
- NetApp Cloud Central account
- 如果使用OCP、则Red Hat OpenShift Container Platform (OCP)权限(在命名空间级别用于创建Pod)
- AWS 凭据，访问 ID 和机密密钥，具有用于创建存储分段和连接器的权限
- AWS 帐户弹性容器注册（ Elastic Container Registry ， ECR ）访问和登录
- 要访问Astra Control UI、需要AWS托管区域和Amazon Route 53条目

AWS 的操作环境要求

Astra 控制中心需要以下 AWS 操作环境：

- Red Hat OpenShift Container Platform 4.11至4.13

确保您选择托管 Astra 控制中心的操作环境满足环境官方文档中概述的基本资源要求。

除了环境的资源要求之外、Asta Control Center还需要特定的资源。请参见 "Astra 控制中心运营环境要求"。

AWS注册表令牌将在12小时后过期、之后您必须续订Docker映像注册表密钥。

AWS 部署概述

下面简要介绍了将 Cloud Volumes ONTAP 作为存储后端安装适用于 AWS 的 Astra 控制中心的过程。

下面详细介绍了其中每个步骤。

1. 确保您具有足够的 IAM 权限。
2. 在 AWS 上安装 RedHat OpenShift 集群。
3. 配置 AWS。
4. 配置适用于AWS的NetApp BlueXP。
5. 安装适用于AWS的Astra控制中心。

确保您具有足够的 IAM 权限

确保您具有足够的IAM角色和权限、可以安装RedHat OpenShift集群和NetApp BlueXP (以前称为Cloud Manager) Connector。

请参见 "初始 AWS 凭据"。

在 AWS 上安装 RedHat OpenShift 集群

在 AWS 上安装 RedHat OpenShift 容器平台集群。

有关安装说明，请参见 "在 OpenShift 容器平台中的 AWS 上安装集群"。

配置 AWS

接下来、将AWS配置为创建虚拟网络、设置EC2计算实例以及创建AWS S3存储分段。如果无法访问NetApp

Astra控制中心映像注册表、则还需要创建一个Elastic Container Registry (ECR)来托管Astra控制中心映像、并将这些映像推送到此注册表。

按照 AWS 文档完成以下步骤。请参见 "AWS 安装文档"。

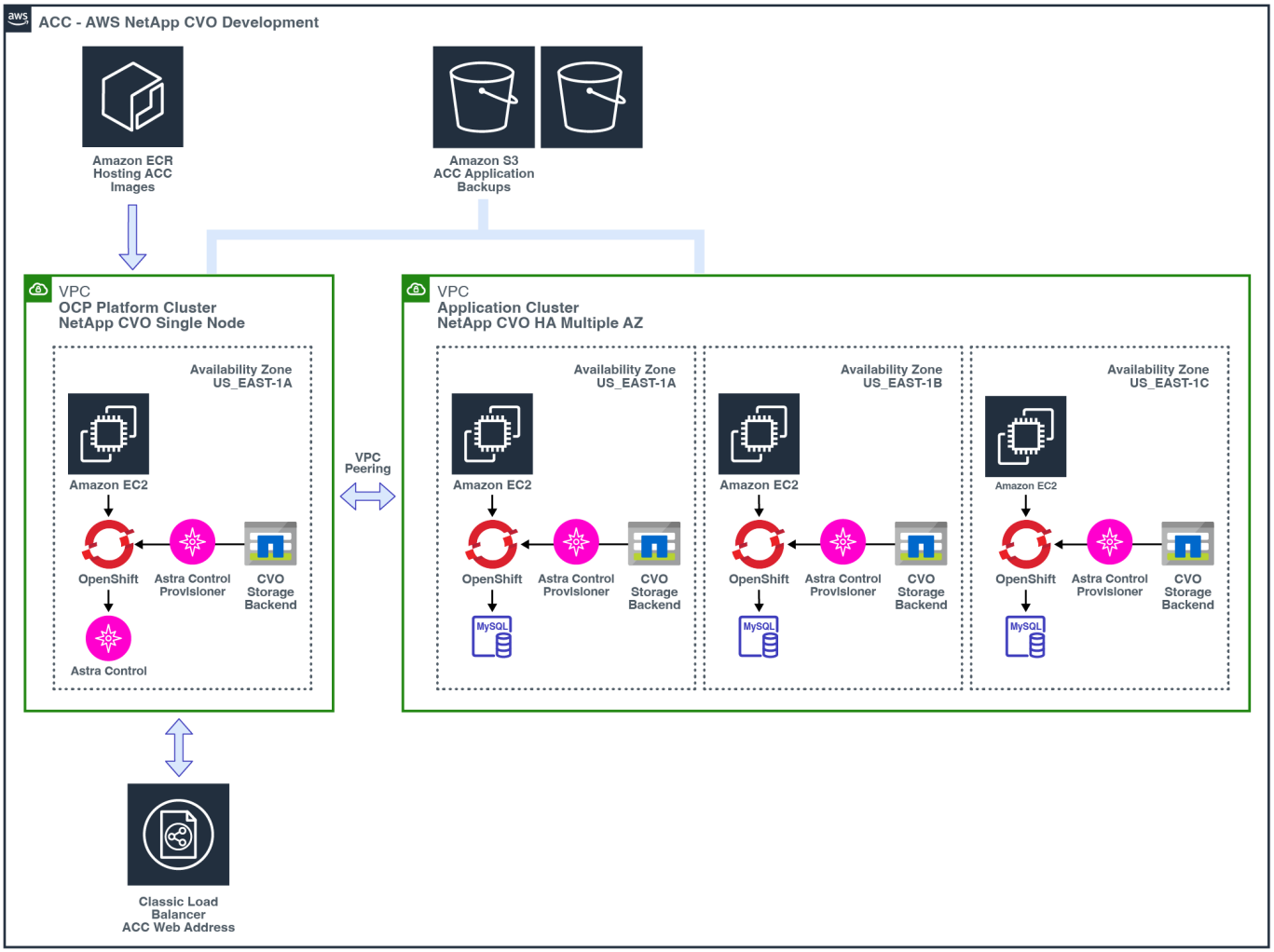
1. 创建AWS虚拟网络。
2. 查看 EC2 计算实例。这可以是 AWS 中的裸机服务器或 VM 。
3. 如果实例类型尚未与主节点和工作节点的 Astra 最低资源要求匹配，请更改 AWS 中的实例类型以满足 Astra 要求。 请参见 "Astra 控制中心要求"。
4. 至少创建一个 AWS S3 存储分段来存储备份。
5. (可选)如果无法访问NetApp映像注册表、请执行以下操作：
   a. 创建AWS Elastic Container Registry (ECR)以托管所有Astra Control Center映像。

      如果不创建ECR、则Astra控制中心无法从包含Cloud Volumes ONTAP 且具有AWS后端的集群访问监控数据。如果您尝试使用 Astra 控制中心发现和管理的集群没有 AWS ECR 访问权限，则会导致出现问题描述 。

   b. 将A作用 力控制中心图像推送到您定义的注册表。

      AWS 弹性容器注册表（ ECR ）令牌将在 12 小时后过期，并导致跨集群克隆操作失败。从为AWS配置的Cloud Volumes ONTAP 管理存储后端时会发生此问题描述。要更正此问题描述 ，请再次向 ECR 进行身份验证，并生成一个新密钥，以便成功恢复克隆操作。

以下是 AWS 部署示例：



配置适用于**AWS**的**NetApp BlueXP**

使用NetApp BlueXP (以前称为Cloud Manager)创建工作空间、向AWS添加连接器、创建工作环境并导入集群。

按照BlueXP文档完成以下步骤。请参见以下内容：

- "AWS 中的 Cloud Volumes ONTAP 入门"。
- "使用BlueXP在AWS中创建连接器"

步骤

1. 将凭据添加到BlueXP。
2. 创建工作空间。
3. 为 AWS 添加连接器。选择 AWS 作为提供程序。
4. 为您的云环境创建一个工作环境。
    a. 位置： "Amazon Web Services （ AWS ） "
    b. 类型： Cloud Volumes ONTAP HA
5. 导入 OpenShift 集群。集群将连接到您刚刚创建的工作环境。
    a. 选择 * K8s* > * 集群列表 * > * 集群详细信息 * ，查看 NetApp 集群详细信息。

b. 请注意右上角的Asta Control配置程序版本。

c. 记下显示 NetApp 作为配置程序的 Cloud Volumes ONTAP 集群存储类。

此操作将导入 Red Hat OpenShift 集群并为其分配默认存储类。您可以选择存储类。Asta Control配置程序会在导入和发现过程中自动安装。

6. 记下此Cloud Volumes ONTAP 部署中的所有永久性卷和卷。

Cloud Volumes ONTAP 可以作为单个节点运行，也可以在高可用性环境下运行。如果已启用 HA ，请记下在 AWS 中运行的 HA 状态和节点部署状态。

安装适用于AWS的Astra控制中心

请遵循标准 "Astra 控制中心安装说明"。

AWS使用通用S3存储分段类型。

在Google Cloud Platform中部署Astra控制中心

您可以在Google云平台(GCP)公有 云上托管的自管理Kubernetes集群上部署Astra控制中心。

GCP所需的功能

在GCP中部署Astra Control Center之前、您需要以下各项：

- Astra Control Center 许可证。请参见 "Astra 控制中心许可要求"。
- "满足 Astra 控制中心的要求"。
- NetApp Cloud Central account
- 如果使用OCP、则为Red Hat OpenShift Container Platform (OCP) 4.11至4.13
- 如果使用OCP、则Red Hat OpenShift Container Platform (OCP)权限(在命名空间级别用于创建Pod)
- GCP服务帐户、具有创建存储分段和连接器的权限

GCP的操作环境要求

确保您选择托管 Astra 控制中心的操作环境满足环境官方文档中概述的基本资源要求。

除了环境的资源要求之外、Asta Control Center还需要特定的资源。请参见 "Astra 控制中心运营环境要求"。

GCP部署概述

下面概述了在GCP中将Cloud Volumes ONTAP 作为存储后端的自管理OCP集群上安装Astra控制中心的过程。

下面详细介绍了其中每个步骤。

1. 在GCP上安装RedHat OpenShift集群。
2. 创建GCP项目和虚拟私有云。
3. 确保您具有足够的 IAM 权限。

4. 配置GCP。
5. 为GCP配置NetApp BlueXP。
6. 安装适用于GCP的Astra控制中心。

在GCP上安装RedHat OpenShift集群

第一步是在GCP上安装RedHat OpenShift集群。

有关安装说明，请参见以下内容：

- "在GCP中安装OpenShift集群"
- "创建GCP服务帐户"

创建GCP项目和虚拟私有云

至少创建一个GCP项目和虚拟私有云(Virtual Private Cloud、VPC)。

OpenShift 可能会创建自己的资源组。此外、您还应定义GCP VPC。请参见 OpenShift 文档。

您可能需要创建平台集群资源组和目标应用程序 OpenShift 集群资源组。

确保您具有足够的 IAM 权限

确保您具有足够的IAM角色和权限、可以安装RedHat OpenShift集群和NetApp BlueXP (以前称为Cloud Manager) Connector。

请参见 "初始GCP凭据和权限"。

配置GCP

接下来、配置GCP以创建VPC、设置计算实例以及创建Google Cloud Object Storage。如果无法访问NetApp Astra控制中心映像注册表、您还需要创建一个Google容器注册表来托管Astra控制中心映像、并将这些映像推送到此注册表。

按照GCP文档完成以下步骤。请参见在GCP中安装OpenShift集群。

1. 在GCP中创建一个GCP项目和VPC、该项目和VPC计划用于具有CVO后端的OCP集群。
2. 查看计算实例。此服务器可以是GCP中的裸机服务器或VM。
3. 如果实例类型尚未与主节点和工作节点的Astra最低资源要求匹配、请在GCP中更改实例类型以满足Astra要求。请参见 "Astra 控制中心要求"。
4. 至少创建一个GCP Cloud Storage Bucket以存储备份。
5. 创建存储分段访问所需的密钥。
6. (可选)如果无法访问NetApp映像注册表、请执行以下操作：
    a. 创建Google容器注册表以托管Asta Control Center映像。
    b. 为所有Astra控制中心映像设置用于Docker推/拉的Google容器注册表访问权限。

示例：可以通过输入以下脚本将Astra Control Center映像推送到此注册表：

```
gcloud auth activate-service-account <service account email address>
--key-file=<GCP Service Account JSON file>
```

此脚本需要一个Astra控制中心清单文件以及您的Google映像注册表位置。示例

```
manifestfile=acc.manifest.bundle.yaml
GCP_CR_REGISTRY=<target GCP image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

7. 设置 DNS 区域。

为GCP配置NetApp BlueXP

使用NetApp BlueXP (以前称为Cloud Manager)创建工作空间、向GCP添加连接器、创建工作环境并导入集群。

按照BlueXP文档完成以下步骤。请参见 "GCP中的Cloud Volumes ONTAP 入门"。

开始之前

- 使用所需的IAM权限和角色访问GCP服务帐户

步骤

1. 将凭据添加到BlueXP。请参见 "正在添加GCP帐户"。
2. 为GCP添加一个连接器。
   a. 选择"GCP"作为提供程序。
   b. 输入GCP凭据。请参见 "从BlueXP在GCP中创建连接器"。
   c. 确保连接器正在运行，然后切换到该连接器。
3. 为您的云环境创建一个工作环境。
   a. 位置："GCP"
   b. 类型： Cloud Volumes ONTAP HA
4. 导入 OpenShift 集群。集群将连接到您刚刚创建的工作环境。
   a. 选择 * K8s* > * 集群列表 * > * 集群详细信息 * ，查看 NetApp 集群详细信息。

b. 请注意右上角的Asta Control配置程序版本。

c. 记下显示为"netapp"作为配置程序的Cloud Volumes ONTAP 集群存储类。

此操作将导入 Red Hat OpenShift 集群并为其分配默认存储类。您可以选择存储类。Asta Control配置程序会在导入和发现过程中自动安装。

5. 记下此Cloud Volumes ONTAP 部署中的所有永久性卷和卷。

Cloud Volumes ONTAP 可以作为单个节点运行、也可以在高可用性(HA)中运行。如果已启用HA、请记下在GCP中运行的HA状态和节点部署状态。

安装适用于**GCP**的**Astra**控制中心

请遵循标准 "Astra 控制中心安装说明"。

GCP使用通用S3存储分段类型。

1. 生成Docker密钥以提取用于Astra控制中心安装的映像：

```
kubectl create secret docker-registry <secret name> --docker
-server=<Registry location> --docker-username=_json_key --docker
-password="$(cat <GCP Service Account JSON file>)" --namespace=pcloud
```

在 **Microsoft Azure** 中部署 **Astra** 控制中心

您可以在 Microsoft Azure 公有 云上托管的自管理 Kubernetes 集群上部署 Astra 控制中心。

Azure所需的功能

在Azure中部署Astra Control Center之前、您需要满足以下条件：

- Astra Control Center 许可证。请参见 "Astra 控制中心许可要求"。
- "满足 Astra 控制中心的要求"。
- NetApp Cloud Central account
- 如果使用OCP、则为Red Hat OpenShift Container Platform (OCP) 4.11至4.13
- 如果使用OCP、则Red Hat OpenShift Container Platform (OCP)权限(在命名空间级别用于创建Pod)
- 具有用于创建存储分段和连接器的权限的 Azure 凭据

Azure 的操作环境要求

确保您选择托管 Astra 控制中心的操作环境满足环境官方文档中概述的基本资源要求。

除了环境的资源要求之外、Asta Control Center还需要特定的资源。请参见 "Astra 控制中心运营环境要求"。

Azure 部署概述

下面简要介绍了适用于 Azure 的 Astra 控制中心的安装过程。

下面详细介绍了其中每个步骤。

1. 在 Azure 上安装 RedHat OpenShift 集群。
2. 创建 Azure 资源组。
3. 确保您具有足够的 IAM 权限。
4. 配置 Azure。
5. 为Azure配置NetApp BlueXP (以前称为Cloud Manager)。
6. 安装和配置适用于Azure的Astra控制中心。

在 Azure 上安装 RedHat OpenShift 集群

第一步是在 Azure 上安装 RedHat OpenShift 集群。

有关安装说明，请参见以下内容：

- "在 Azure 上安装 OpenShift 集群"。
- "安装 Azure 帐户"。

创建 Azure 资源组

至少创建一个 Azure 资源组。

> OpenShift 可能会创建自己的资源组。除了这些之外，您还应定义 Azure 资源组。请参见 OpenShift 文档。

您可能需要创建平台集群资源组和目标应用程序 OpenShift 集群资源组。

确保您具有足够的 IAM 权限

确保您具有足够的IAM角色和权限、可以安装RedHat OpenShift集群和NetApp BlueXP Connector。

请参见 "Azure 凭据和权限"。

配置 Azure

接下来、将Azure配置为创建虚拟网络、设置计算实例以及创建Azure Blb容器。如果您无法访问NetApp Astra控制中心映像注册表、则还需要创建Azure容器注册表(ACR)来托管Astra控制中心映像、并将这些映像推送到此注册表。

按照 Azure 文档完成以下步骤。请参见 "在 Azure 上安装 OpenShift 集群"。

1. 创建Azure虚拟网络。
2. 查看计算实例。这可以是 Azure 中的裸机服务器或 VM 。
3. 如果实例类型尚未与主节点和工作节点的 Astra 最低资源要求匹配，请在 Azure 中更改实例类型以满足

Astra 要求。 请参见 "Astra 控制中心要求"。

4. 至少创建一个Azure Blob容器以存储备份。
5. 创建存储帐户。您需要使用存储帐户来创建要在Astra Control Center中用作存储分段的容器。
6. 创建存储分段访问所需的密钥。
7. (可选)如果无法访问NetApp映像注册表、请执行以下操作：

   a. 创建Azure容器注册表(ACR)以托管Asta控制中心映像。

   b. 为所有Astra Control Center映像设置Docker推送/拉取的ACR访问权限。

   c. 使用以下脚本将Astra Control Center映像推送到此注册表：

```
az acr login -n <AZ ACR URL/Location>
This script requires the Astra Control Center manifest file and your
Azure ACR location.
```

   - 示例 * ：

```
manifestfile=acc.manifest.bundle.yaml
AZ_ACR_REGISTRY=<target Azure ACR image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < acc.manifest.bundle.yaml
```

8. 设置 DNS 区域。

为Azure配置NetApp BlueXP (以前称为Cloud Manager)

使用BlueXP (以前称为Cloud Manager)创建工作空间、向Azure添加连接器、创建工作环境并导入集群。

按照BlueXP文档完成以下步骤。请参见 "Azure中的BlueXP入门"。

开始之前

使用所需的 IAM 权限和角色访问 Azure 帐户

步骤

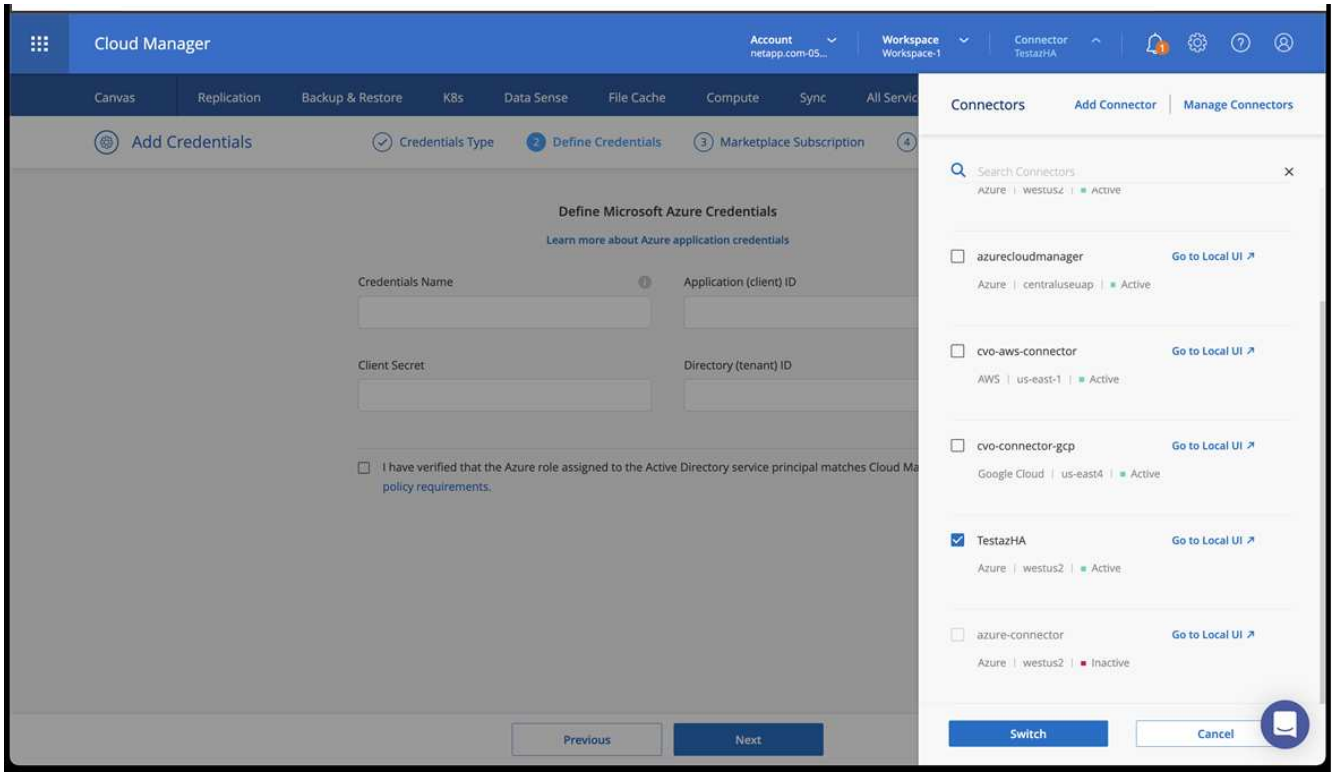
1. 将凭据添加到BlueXP。
2. 添加适用于 Azure 的连接器。请参见 "BlueXP策略"。

a. 选择 * Azure * 作为提供程序。

b. 输入 Azure 凭据，包括应用程序 ID ，客户端密钥和目录（租户） ID 。

请参见 "从BlueXPr.在Azure中创建连接器"。

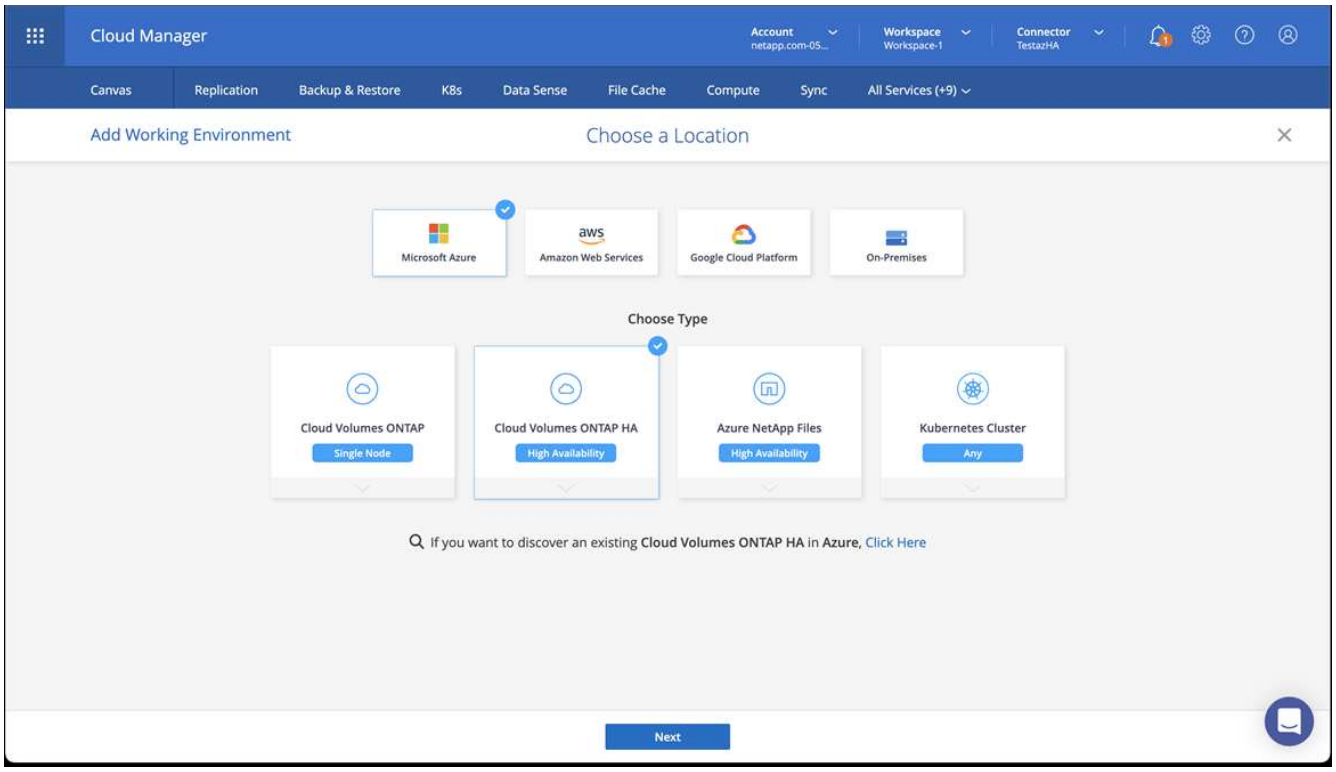
3. 确保连接器正在运行，然后切换到该连接器。



4. 为您的云环境创建一个工作环境。

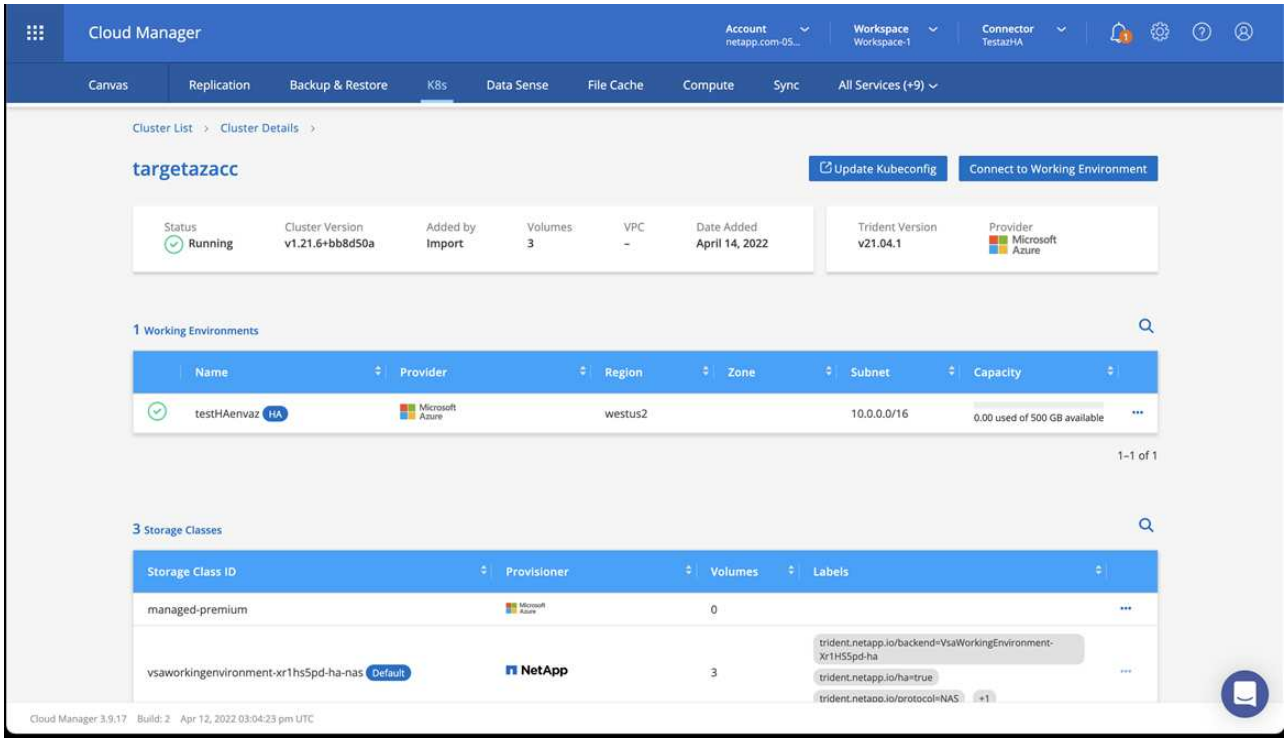
a. 位置： "Microsoft Azure" 。

b. 键入： Cloud Volumes ONTAP HA 。



5. 导入 OpenShift 集群。集群将连接到您刚刚创建的工作环境。

   a. 选择 * K8s* > * 集群列表 * > * 集群详细信息 * ，查看 NetApp 集群详细信息。



   b. 请注意右上角的Asta Control配置程序版本。

   c. 记下显示 NetApp 作为配置程序的 Cloud Volumes ONTAP 集群存储类。

此操作将导入 Red Hat OpenShift 集群并分配默认存储类。您可以选择存储类。

Asta Control配置程序会在导入和发现过程中自动安装。

6. 记下此Cloud Volumes ONTAP 部署中的所有永久性卷和卷。
7. Cloud Volumes ONTAP 可以作为单个节点运行，也可以在高可用性环境下运行。如果已启用 HA ，请记下在 Azure 中运行的 HA 状态和节点部署状态。

安装和配置适用于**Azure**的**Astra**控制中心

按照标准安装 Astra 控制中心 "安装说明"。

使用 Astra 控制中心添加 Azure 存储分段。请参见 "设置 Astra 控制中心并添加存储分段"。

安装后配置**Astra**控制中心

根据您的环境、安装Astra控制中心后可能需要进行其他配置。

消除资源限制

某些环境使用ResourceQuotas和LimitRanges对象来防止命名空间中的资源占用集群上的所有可用CPU和内存。Astra控制中心未设置最大限制、因此不符合这些资源的要求。如果您的环境采用这种方式配置、则需要从计划安装Astra控制中心的命名空间中删除这些资源。

您可以使用以下步骤检索和删除这些配额和限制。在这些示例中、命令输出会立即显示在命令后面。

步骤

1. 在中获取资源配额 `netapp-acc` (或自定义名称)命名空间：

```
kubectl get quota -n [netapp-acc or custom namespace]
```

响应：

```
NAME          AGE   REQUEST                                              LIMIT
pods-high     16s   requests.cpu: 0/20, requests.memory: 0/100Gi
limits.cpu: 0/200, limits.memory: 0/1000Gi
pods-low      15s   requests.cpu: 0/1, requests.memory: 0/1Gi
limits.cpu: 0/2, limits.memory: 0/2Gi
pods-medium   16s   requests.cpu: 0/10, requests.memory: 0/20Gi
limits.cpu: 0/20, limits.memory: 0/200Gi
```

2. 按名称删除所有资源配额：

```
kubectl delete resourcequota pods-high -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. 在中获取限制范围 `netapp-acc` (或自定义名称)命名空间：

```
kubectl get limits -n [netapp-acc or custom namespace]
```

响应：

```
NAME              CREATED AT
cpu-limit-range   2022-06-27T19:01:23Z
```

4. 按名称删除限制范围：

```
kubectl delete limitrange cpu-limit-range -n [netapp-acc or custom
namespace]
```

添加自定义 **TLS** 证书

默认情况下、Astra控制中心对传入控制器流量(仅在某些配置中)和Web浏览器的Web UI身份验证使用自签名TLS证书。对于生产环境、您应删除现有的自签名TLS证书、并将其替换为由证书颁发机构(CA)签名的TLS证书。

> 默认的自签名证书用于两种类型的连接：
>
> - 通过HTTPS连接到Astra控制中心Web UI
> - 传入控制器流量(仅当 `ingressType: "AccTraefik"` 属性已在中设置 `astra_control_center.yaml` 在安装Astra Control Center期间生成文件)
>
> 替换默认TLS证书将替换用于对这些连接进行身份验证的证书。

开始之前

- 安装了 Astra 控制中心的 Kubernetes 集群
- 对集群上的命令 Shell 进行管理访问，以运行 `kubectl` 命令
- CA 中的专用密钥和证书文件

删除自签名证书

删除现有的自签名 TLS 证书。

1. 使用 SSH ，以管理用户身份登录到托管 Astra 控制中心的 Kubernetes 集群。
2. 使用以下命令查找与当前证书关联的 TLS 密钥，并将 ` <Acc-deployment-namespace>` 替换为 Astra Control Center 部署命名空间：

```
kubectl get certificate -n <ACC-deployment-namespace>
```

3. 使用以下命令删除当前安装的密钥和证书：

```
kubectl delete cert cert-manager-certificates -n <ACC-deployment-namespace>
```

```
kubectl delete secret secure-testing-cert -n <ACC-deployment-namespace>
```

使用命令行添加新证书

添加一个由 CA 签名的新 TLS 证书。

1. 使用以下命令使用 CA 中的专用密钥和证书文件创建新的 TLS 密钥，并将括号 <> 中的参数替换为相应的信息：

```
kubectl create secret tls <secret-name> --key <private-key-filename> --cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. 使用以下命令和示例编辑集群自定义资源定义（ CRD ）文件，并将 `spec.selfSigned` 值更改为 `spec.ca.secretName` ，以引用您先前创建的 TLS 密钥：

```
kubectl edit clusterissuers.cert-manager.io/cert-manager-certificates -n <ACC-deployment-namespace>
```

CRD：

```
#spec:
#  selfSigned: {}

spec:
  ca:
    secretName: <secret-name>
```

3. 使用以下命令和示例输出验证所做的更改是否正确以及集群是否已准备好验证证书，并将 ` <Acc-deployment-namespace>` 替换为 Astra Control Center 部署命名空间：

```
kubectl describe clusterissuers.cert-manager.io/cert-manager-certificates -n <ACC-deployment-namespace>
```

响应：

```
Status:
  Conditions:
    Last Transition Time:  2021-07-01T23:50:27Z
    Message:               Signing CA verified
    Reason:                KeyPairVerified
    Status:                True
    Type:                  Ready
Events:                    <none>
```

4. 使用以下示例创建 `certificate.yaml` 文件，将括号中的占位值替换为相应的信息：

> 此示例使用 `dnsNames` 属性以指定Asta Control Center DNS地址。Astra Control Center不支持使用公用名(Common Name、CN)属性指定DNS地址。

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  <strong>name: <certificate-name></strong>
  namespace: <ACC-deployment-namespace>
spec:
  <strong>secretName: <certificate-secret-name></strong>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  <strong>- <astra.dnsname.example.com></strong> #Replace with the correct Astra Control Center DNS address
  issuerRef:
    kind: ClusterIssuer
    name: cert-manager-certificates
```

5. 使用以下命令创建证书：

```
kubectl apply -f certificate.yaml
```

6. 使用以下命令和示例输出，验证是否已正确创建证书以及是否使用您在创建期间指定的参数（例如名称，持续时间，续订截止日期和 DNS 名称）。

```
kubectl describe certificate -n <ACC-deployment-namespace>
```

响应：

```
Spec:
  Dns Names:
    astra.example.com
  Duration:  125h0m0s
  Issuer Ref:
    Kind:         ClusterIssuer
    Name:         cert-manager-certificates
  Renew Before:  61h0m0s
  Secret Name:   <certificate-secret-name>
Status:
  Conditions:
    Last Transition Time:  2021-07-02T00:45:41Z
    Message:               Certificate is up to date and has not expired
    Reason:                Ready
    Status:                True
    Type:                  Ready
  Not After:               2021-07-07T05:45:41Z
  Not Before:              2021-07-02T00:45:41Z
  Renewal Time:            2021-07-04T16:45:41Z
  Revision:                1
Events:                    <none>
```

7. 使用以下命令和示例编辑TLS存储CRD以指向新证书密钥名称、并将括号<>中的占位符值替换为适当的信息

```
kubectl edit tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD：

```
...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>
```

8. 使用以下命令和示例编辑传入 CRD TLS 选项以指向新的证书密钥，并将括号 <> 中的占位符值替换为相应的信息：

```
kubectl edit ingressroutes.traefik.io -n <ACC-deployment-namespace>
```

CRD：

```
...
 tls:
    secretName: <certificate-secret-name>
```

9. 使用 Web 浏览器浏览到 Astra 控制中心的部署 IP 地址。
10. 验证证书详细信息是否与您安装的证书的详细信息匹配。
11. 导出证书并将结果导入到 Web 浏览器中的证书管理器中。

设置 Astra 控制中心

添加 Astra 控制中心的许可证

安装Astra Control Center时、已安装嵌入式评估版许可证。如果您正在评估Astra Control Center、则可以跳过此步骤。

您可以使用Astra Control UI或添加新许可证 "Astra Control API"。

Astra控制中心许可证使用Kubernetes CPU单元测量CPU资源、并计算分配给所有受管Kubernetes集群的工作节点的CPU资源。许可证基于vCPU使用量。有关如何计算许可证的详细信息、请参见 "许可"。

如果您的安装增长到超过许可的 CPU 单元数，则 Astra 控制中心将阻止您管理新应用程序。超过容量时，将显示警报。

要更新现有评估版或完整许可证、请参见 "更新现有许可证"。

开始之前

- 访问新安装的Astra Control Center实例。
- 管理员角色权限。
- 答 "NetApp 许可证文件" (nlf)。

步骤

1. 登录到 Astra 控制中心 UI 。
2. 选择 * 帐户 * > * 许可证 * 。
3. 选择 * 添加许可证 * 。
4. 浏览到您下载的许可证文件（ NLF ）。
5. 选择 * 添加许可证 * 。

◦ 帐户 * > * 许可证 * 页面显示许可证信息，到期日期，许可证序列号，帐户 ID 和使用的 CPU 单元。

如果您拥有评估版许可证、并且不向AutoSupport 发送数据、请确保存储您的帐户ID、以避免在Astra控制中心发生故障时丢失数据。

启用**Asta Control**配置程序

Astra Trident 23.10及更高版本提供了使用Astra Control配置程序的选项、允许获得许可的Astra Control用户访问高级存储配置功能。除了基于标准Asta三端CSI的功能之外、Asta Control配置程序还提供了此扩展功能。

在即将推出的Astra Control更新中、Astra Control配置程序将取代Astra Trandent作为存储配置程序和流程编排程序、并且Astra Control必须使用它。因此、强烈建议Asta Control用户启用Asta Control配置程序。Asta三元数据将继续保持开源状态、并使用NetApp的新CSI和其他功能进行发布、维护、支持和更新。

关于此任务

如果您是Astra控制中心的许可用户、并且希望使用Astra控制配置程序功能、则应遵循此操作步骤。如果您是Astra三端数据库的用户、并且希望使用Astra Control配置程序提供的其他功能、而不同时使用Astra Control、则还应遵循此操作步骤。

对于每种情况、默认情况下在Astra Trident 24.02中不会启用配置程序功能、必须启用此功能。

开始之前

如果要启用Asta Control配置程序、请先执行以下操作：

Asta Control为用户提供Asta Control Center

- 获取**Astra**控制中心许可证：您需要 "Asta Control Center许可证" 启用Astra Control配置程序并访问它提供的功能。
- 安装或升级到**Astra Control Center 23.10**或更高版本：如果您计划在Astra Control中使用最新的Astra Control配置程序功能(24.02)，则需要最新的Astra Control Center版本(24.02)。
- *确认集群具有一个AMD64*系统架构：Astra Control配置程序映像在amd64和ARM64 CPU架构中都提供，但Astra Control Center仅支持amd64。
- 获取用于注册表访问的**Asta Control**服务帐户：如果要使用Asta Control注册表而不是NetApp 支持站点 来下载Asta Control配置程序映像、请完成的注册 "Asta Control Service帐户"。填写并提交表单并创建BlueXP帐户后、您将收到Astra Control Service欢迎电子邮件。
- 如果您安装了**Astra**三端安装程序，请确认其版本在四个版本的窗口内：如果您的Astra三端安装程序在版本24.02的四个版本窗口内，则可以使用Astra Control置备程序直接升级到Astra三端安装程序24.02。例如、您可以直接从Asta三端23.04升级到24.02。

仅适用于**Asta Control**配置程序用户

- 获取**Astra**控制中心许可证：您需要 "Asta Control Center许可证" 启用Astra Control配置程序并访问它提供的功能。
- 如果您安装了**Astra**三端安装程序，请确认其版本在四个版本的窗口内：如果您的Astra三端安装程序在版本24.02的四个版本窗口内，则可以使用Astra Control置备程序直接升级到Astra三端安装程序24.02。例如、您可以直接从Asta三端23.04升级到24.02。
- 获取**Astra Control Service**帐户以访问注册表：您需要访问注册表才能下载Astra Control配置程序映像。要开始使用、请完成的注册 "Asta Control Service帐户"。填写并提交表单并创建BlueXP帐户后、您将收到Astra Control Service欢迎电子邮件。

(步骤1)获取Asta Control配置程序映像

Asta控制中心用户可以使用Asta控制注册表或NetApp 支持站点 方法获取Asta控制配置程序映像。要在不使用Astra Control的情况下使用Astra Control配置程序的Astra Trent用户应使用注册表方法。

Astra Control图像注册表

在此操作步骤中、您可以使用Podman而不是Docker执行命令。如果您使用的是Windows环境、建议使用PowerShell。

1. 访问NetApp Astra控件映像注册表：
   a. 登录到Astra Control Service Web UI、然后选择页面右上角的图图标。
   b. 选择* API访问*。
   c. 记下您的帐户ID。
   d. 在同一页面中，选择*Generate API t令牌*并将API令牌字符串复制到剪贴板，然后将其保存在编辑器中。
   e. 使用您的首选方法登录Astra Control注册表：

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (仅限自定义注册表)按照以下步骤将图像移动到自定义注册表。如果您不使用注册表、请按照中的三端修复操作符步骤进行操作 "下一节"。
   a. 从注册表中提取Asta Control配置程序映像：

提取的映像不支持多个平台、只支持与提取映像的主机相同的平台、例如Linux amd64。

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform <cluster platform>
```

示例

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0 --platform linux/amd64
```

   a. 标记图像：

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0
```

b. 将映像推送到自定义注册表：

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

> 您可以使用"删除副本"来替代运行以下Docker命令：
> `crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0 <my_custom_registry>/trident-acp:24.02.0`

NetApp 支持站点

1. 下载Asta Control配置程序包 (`trident-acp-[version].tar`) "Astra Control Center下载页面"。
2. (建议但可选)下载Astra Control Center的证书和签名捆绑包(astra-control-crier-certs -[version].tar.gz)、以验证trident - acp-[version] tar捆绑包的签名。

```
tar -vxzf astra-control-center-certs-[version].tar.gz
```

```
openssl dgst -sha256 -verify certs/AstraControlCenterDockerImages-public.pub -signature certs/trident-acp-[version].tar.sig trident-acp-[version].tar
```

3. 加载Asta Control配置程序映像：

```
docker load < trident-acp-24.02.0.tar
```

响应：

```
Loaded image: trident-acp:24.02.0-linux-amd64
```

4. 标记图像：

```
docker tag trident-acp:24.02.0-linux-amd64 <my_custom_registry>/trident-acp:24.02.0
```

5. 将映像推送到自定义注册表：

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

(第2步)在Asta Trident中启用Asta Control配置程序

确定原始安装方法是否使用 "运算符(手动或使用Helm)或trdentcd" 并根据原始方法完成相应的步骤。

Asta三端操作员

1. "下载Asta三端安装程序并解压缩"。
2. 如果您尚未安装Astra三端安装程序、或者您从初始Astra三端安装程序中删除了操作员、请完成以下步骤：

   a. 创建客户需求日：

   ```
   kubectl create -f
   deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.y
   aml
   ```

   b. 创建三项命名空间 (`kubectl create namespace trident`)或确认三项命名空间仍然存在 (`kubectl get all -n trident`）。如果已删除此命名空间、请重新创建它。

3. 将Astra Trdent更新到24.02.0：

   对于运行Kubornetes 1.24或更早版本的集群、请使用 `bundle_pre_1_25.yaml`。对于运行Kubernetes 1.25或更高版本的集群、请使用 `bundle_post_1_25.yaml`。

   ```
   kubectl -n trident apply -f trident-installer/deploy/<bundle-
   name.yaml>
   ```

4. 验证Astra trident是否正在运行：

   ```
   kubectl get torc -n trident
   ```

   响应：

   ```
   NAME      AGE
   trident   21m
   ```

5. 如果您有一个使用机密的注册表，请创建一个用于提取Astra Control置备程序映像的密钥：

   ```
   kubectl create secret docker-registry <secret_name> -n trident
   --docker-server=<my_custom_registry> --docker-username=<username>
   --docker-password=<token>
   ```

6. 编辑TridentOrchestrator CR并进行以下编辑：

   ```
   kubectl edit torc trident -n trident
   ```

a. 为Astra三端映像设置自定义注册表位置或从Astra Control注册表中提取该映像 (`tridentImage: <my_custom_registry>/trident:24.02.0` 或 `tridentImage: netapp/trident:24.02.0`) 。

b. 启用Asta Control配置程序 (`enableACP: true`) 。

c. 设置Asta Control配置程序映像的自定义注册表位置或将其从Asta Control注册表中提取 (`acpImage: <my_custom_registry>/trident-acp:24.02.0` 或 `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`) 。

d. 如果您已建立 图像拉取密钥 在本操作步骤的前面部分、您可以在此处设置它们 (`imagePullSecrets: - <secret_name>`) 。使用您在前面步骤中创建的相同名称机密名称。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
  - <secret_name>
```

7. 保存并退出文件。部署过程将自动开始。

8. 验证是否已创建操作员、部署和副本集。

```
kubectl get all -n trident
```

> 在 Kubernetes 集群中只能有 * 一个操作符实例 * 。请勿创建多个部署的Asta三端操作员。

9. 验证 `trident-acp` 容器正在运行 `acpVersion` 为 `24.02.0` 状态为 `Installed`：

```
kubectl get torc -o yaml
```

响应：

```
status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
  status: Installed
```

Tridentctl

1. "下载Asta三端安装程序并解压缩"。
2. "如果您已有Asta Trident、请从托管它的集群中将其卸载"。
3. 在启用Asta Control配置程序的情况下安装Asta Trent (`--enable-acp=true`）：

```
./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02
```

4. 确认已启用Asta Control配置程序：

```
./tridentctl -n trident version
```

响应：

```
+----------------+----------------+-------------+ | SERVER VERSION |
CLIENT VERSION | ACP VERSION | +----------------+----------------
+-------------+ | 24.02.0 | 24.02.0 | 24.02.0. | +----------------
+----------------+-------------+
```

掌舵

1. 如果您安装了Astra Trident 23.07.1或更早版本、 "卸载" 操作员和其他组件。
2. 如果您的Kubornetes集群运行的是1.24或更早版本、请删除PSP：

```
kubectl delete psp tridentoperatorpod
```

3. 添加Astra Trident Helm存储库：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

4. 更新Helm图表：

```
helm repo update netapp-trident
```

响应：

```
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. □Happy Helming!□
```

5. 列出图像：

```
./tridentctl images -n trident
```

响应：

```
| v1.28.0            | netapp/trident:24.02.0|
|                    | docker.io/netapp/trident-autosupport:24.02|
|                    | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                    | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                    | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                    | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                    | registry.k8s.io/sig-storage/csi-node-driver-
registrar:v2.10.0 |
|                    | netapp/trident-operator:24.02.0 (optional)
```

6. 确保提供了三项运算符24.02.0：

```
helm search repo netapp-trident/trident-operator --versions
```

响应：

```
NAME                                CHART VERSION   APP VERSION
DESCRIPTION
netapp-trident/trident-operator 100.2402.0      24.02.0         A
```

7. 使用 ... `helm install` 并运行以下选项之一、其中包括这些设置：
   - 部署位置的名称
   - Astra三端版本
   - Asta Control配置程序映像的名称
   - 用于启用配置程序的标志
   - (可选)本地注册表路径。如果您使用的是本地注册表、则为 " {\f270 {\f151 {\f270} " 可以位于一个注册表或不同的注册表中、但所有CSI映像都必须位于同一注册表中。
   - 三端名称空间

选项

- 没有注册表的映像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-acp:24.02.0
--set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- 一个或多个注册表中的图像

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

您可以使用 `helm list` 查看安装详细信息、例如名称、命名空间、图表、状态、应用程序版本、和修订版号。

如果您在使用Helm部署TRIdent时遇到任何问题、请运行此命令以完全卸载Asta TRIdent：

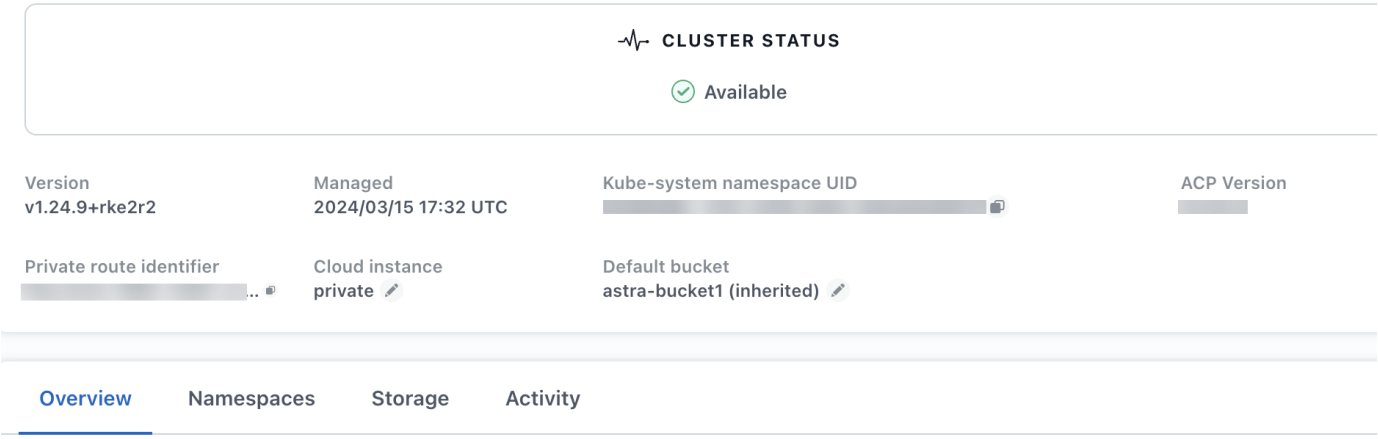
```
./tridentctl uninstall -n trident
```

请勿 "完全删除Asta Trdent CRD" 在尝试重新启用Astra Control配置程序之前、作为卸载的一部分。

结果

Asta Control配置程序功能已启用、您可以使用当前运行的版本可用的任何功能。

(仅适用于Asta Control Center用户)安装Asta Control配置程序后、在Asta Control Center UI中托管此配置程序的集群将显示 `ACP version` 而不是 `Trident version` 字段和当前安装的版本号。



有关详细信息 ...

- "Asta Trdent升级文档"

使用Astra Control准备用于集群管理的环境

在添加集群之前、应确保满足以下前提条件。此外、您还应运行资格检查、以确保您的集群已准备好添加到Astra Control Center、并根据需要创建kubeconfig"集群角色。

Astra Control允许您根据环境和首选项添加由自定义资源(Custom Resource、CR)或kubeconfig"管理的集群。

开始之前

- 满足环境前提条件：您的环境满足 "操作环境要求" A作用 控制中心。
- 配置工作节点：确保您 "配置工作节点" 在集群中使用适当的存储驱动程序、以便Pod可以与后端存储进行交互。
- 启用**PSA**限制：如果集群启用了POD安全准入强制(这是Kubernetes 1.25及更高版本集群的标准配置)、则需要对以下名称空间启用PSA限制：
  - `netapp-acc-operator` 命名空间：

```
kubectl label --overwrite ns netapp-acc-operator pod-
security.kubernetes.io/enforce=privileged
```

  - `netapp monitoring` 命名空间：

```
kubectl label --overwrite ns netapp-monitoring pod-
security.kubernetes.io/enforce=privileged
```

- * ONTAP 凭据*：您需要在备用ONTAP 系统上设置ONTAP 凭据以及超级用户和用户ID、以便使用Astra控制中心备份和还原应用程序。

  在ONTAP 命令行中运行以下命令：

```
export-policy rule modify -vserver <storage virtual machine name>
-policyname <policy name> -ruleindex 1 -superuser sys
export-policy rule modify -vserver <storage virtual machine name>
-policyname <policy name> -ruleindex 1 -anon 65534
```

- *kubeconfig-managed cluster requirement *：这些要求特定于由kubeconfig-managed的应用程序集群。
  - 使**kubeconfig***可访问：您可以访问 "默认集群kubeconfig" 那 "您在安装期间配置的"。
  - 证书颁发机构注意事项：如果使用引用私有证书颁发机构(CA)的kubeconfigfile文件添加集群、请将以下行添加到 `cluster` kubeconfig"文件的部分。这样、Astra Control便可添加集群：

```
insecure-skip-tls-verify: true
```

  - *仅Rancher *：在Rancher环境中管理应用程序集群时、请修改Rancher提供的kubeconfig文件中的应用程序集群默认上下文、以使用控制平面上下文、而不是Rancher API服务器上下文。这样可以减少Rancher API 服务器上的负载并提高性能。
- **Astra Control**配置程序要求：要管理集群、您应正确配置Astra Control配置程序(包括其Astra三项功能组件)。
  - 查看**Astra**三端环境要求：在安装或升级Astra Control配置程序之前、请查看 "支持的前端、后端和主机配置"。
  - 启用**Astra Control**配置程序功能：强烈建议您安装Astra Trident 23.10或更高版本并启用 "Asta Control配置程序高级存储功能"。在未来版本中、如果Asta Control配置程序未启用、则Asta Control将不支持Asta Trent。
  - 配置存储后端：必须至少有一个存储后端 "已在Asta Trdent中配置" 在集群上。
  - 配置存储类：必须至少有一个存储类 "已在Asta Trdent中配置" 在集群上。如果配置了默认存储类，请确保该存储类是具有默认标注的*Only"存储类。
  - 配置卷快照控制器并安装卷快照类： "安装卷快照控制器" 以便可以在Astra Control中创建快照。 "创建" 至少一个 `VolumeSnapshotClass` 使用Asta三端到功能。

运行资格检查

运行以下资格检查，以确保您的集群已准备好添加到 Astra 控制中心。

步骤

1. 确定您正在运行的Astra三项目标版本：

```
kubectl get tridentversion -n trident
```

如果存在Asta三项功能、您将看到类似于以下内容的输出：

```
NAME      VERSION
trident   24.02.0
```

如果Astra三端存储不存在、则会显示类似于以下内容的输出：

```
error: the server doesn't have a resource type "tridentversions"
```

2. 执行以下操作之一：
   - 如果您运行的是Asta三端凹凸版23.01或更早版本、请使用这些版本 "说明" 在升级到Asta Control配置程序之前、升级到Asta三端到最新版本。您可以 "执行直接升级" 如果您的Astra三端存储在版本24.02的四个版本的窗口中、则将Astra Control配置程序更新为24.02。例如、您可以直接从Asta三端23.04升级到Asta Control配置程序24.02。
   - 如果您运行的是Astra Trident 23.10或更高版本、请验证Astra Control配置程序是否已启用 "enabled" 。Asta Control配置程序不能用于23.10之前的Asta Control Center版本。 "升级Astra Control配置程序" 以便它与您要升级的Astra Control Center版本相同、以访问最新功能。
3. 确保所有Pod (包括 `trident-acp`)正在运行：

```
kubectl get pods -n trident
```

4. 确定存储类是否正在使用受支持的Asta三端驱动程序。配置程序名称应为 `csi.trident.netapp.io`。请参见以下示例：

```
kubectl get sc
```

响应示例：

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io   Delete          Immediate
true                    5d23h
```

创建集群角色**kubeconfig**

对于使用kubeconfig"管理的集群、您可以选择为Astra Control Center创建有限权限或扩展权限管理员角色。这不是Astra控制中心设置所需的操作步骤、因为您已在中配置了kubeconfig "安装过程"。

如果您适用场景的环境发生以下任一情况、则此操作步骤可帮助您创建一个单独的kubeconfig:

- 您希望限制Astra Control对其管理的集群的权限
- 您使用多个环境、并且不能使用在安装期间配置的默认Asta Control kubeconfig,否则在您的环境中使用单一环境的有限角色将不起作用

开始之前

在完成操作步骤 步骤之前、请确保您对要管理的集群具有以下信息：

- 已安装kubectl v1.23或更高版本
- kubectl访问要使用Astra控制中心添加和管理的集群

> 对于此操作步骤 、您不需要对运行Astra控制中心的集群进行kubectl访问。

- 要使用活动环境的集群管理员权限管理的集群的活动kubeconfig

步骤

1. 创建服务帐户：
   a. 创建名为`asacontrol service-account.yaml`的服务帐户文件。

```
<strong>astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

   b. 应用服务帐户：

```
kubectl apply -f astracontrol-service-account.yaml
```

2. 创建以下具有足够权限的集群角色之一、以使集群由Astra Control管理：

集群角色受限

此角色包含由Asta Control管理集群所需的最低权限：

a. 创建 `ClusterRole` 文件、例如、 `astra-admin-account.yaml`。

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
  - daemonsets
  - deployments
```

```
    - horizontalpodautoscalers
    - ingresses
    - jobs
    - namespaces
    - networkpolicies
    - persistentvolumeclaims
    - poddisruptionbudgets
    - pods
    - podtemplates
    - replicasets
    - replicationcontrollers
    - replicationcontrollers/scale
    - rolebindings
    - roles
    - secrets
    - serviceaccounts
    - services
    - statefulsets
    - tridentmirrorrelationships
    - tridentsnapshotinfos
    - volumesnapshots
    - volumesnapshotcontents
    verbs:
    - delete

  # Watch resources
  # Necessary to monitor progress
  - apiGroups:
    - ""
    resources:
    - pods
    - replicationcontrollers
    - replicationcontrollers/scale
    verbs:
    - watch

  # Update resources
  - apiGroups:
    - ""
    - build.openshift.io
    - image.openshift.io
    resources:
    - builds/details
    - replicationcontrollers
    - replicationcontrollers/scale
    - imagestreams/layers
```

```
  - imagestreamtags
  - imagetags
  verbs:
  - update
```

b. (仅适用于OpenShift集群)在末尾附加以下内容 `astra-admin-account.yaml` 文件：

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
  - update
```

c. 应用集群角色：

```
kubectl apply -f astra-admin-account.yaml
```

已扩展集群角色

此角色包含要由Asta Control管理的集群的扩展权限。如果您使用多个环境，并且无法使用在安装期间配置的默认Asta Control kubeconfig,则可以使用此角色，否则在您的环境中，只使用一个环境的有限角色将不起作用：

以下内容 `ClusterRole` 步骤是一个常规Kubbernetes示例。有关特定于您的环境的说明、请参见Kubennetes分发版的文档。

a. 创建 `ClusterRole` 文件、例如、 `astra-admin-account.yaml`。

```
<strong>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

b. 应用集群角色：

```
kubectl apply -f astra-admin-account.yaml
```

3. 为集群角色创建与服务帐户的集群角色绑定：

a. 创建一个 ClusterRoleBindingm 文件，该文件名为 astracontrol – clusterrolebind.YAML 。

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: astra-admin-account
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

b. 应用集群角色绑定：

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

4. 创建并应用令牌密钥：

   a. 创建名为的令牌机密文件 `secret-astracontrol-service-account.yaml`。

```
<strong>secret-astracontrol-service-account.yaml</strong>
```

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

   b. 应用令牌密钥：

```
kubectl apply -f secret-astracontrol-service-account.yaml
```

5. 通过将令牌密钥名称添加到、将其添加到服务帐户 `secrets` 数组(以下示例中的最后一行)：

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. 列出服务帐户密码，将 ` <context>` 替换为适用于您的安装的正确上下文：

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

输出的结尾应类似于以下内容：

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

中每个元素的索引 `secrets` 阵列以0开头。在上面的示例中、是的索引 `astracontrol-service-account-dockercfg-48xhx` 将为0、并为创建索引 `secret-astracontrol-service-account` 将为1。在输出中、记下服务帐户密钥的索引编号。在下一步中、您将需要此索引编号。

7. 按如下所示生成 kubeconfig：
   a. 创建 `create-kubeconfig.sh` 文件
   b. 替换 `TOKEN_INDEX` 在以下脚本的开头、使用正确的值。

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')

TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
```

```
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token-
user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

c. 获取用于将其应用于 Kubernetes 集群的命令。

```
source create-kubeconfig.sh
```

8. (可选)将kubeconfig重命名为集群的有意义名称。

```
mv kubeconfig-sa YOUR_CLUSTER_NAME_kubeconfig
```

(技术预览)为受管集群安装Asta Connector

由Asta Control Center管理的集群使用Asta Connector在受管集群和Asta Control Center之间实现通信。您需要在要管理的所有集群上安装Astra Connector。

安装A作用 连接器

您可以使用Kubbernetes命令和自定义资源(Custom Resource、CR)文件安装Astra Connector。

关于此任务

- 执行这些步骤时、请在要使用Astra Control进行管理的集群上执行这些命令。
- 如果使用的是Bastion主机、请从Bastion主机的命令行对这些命令执行问题描述。

开始之前

- 您需要访问要使用Astra Control管理的集群。
- 要在集群上安装Asta Connector操作员、您需要具有Kubbernetes管理员权限。

> 如果为集群配置了强制实施Pod安全接入(这是Kubernetes 1.25及更高版本集群的默认设置)、则需要对相应的卷空间启用PSA限制。请参见 "使用Astra Control准备用于集群管理的环境" 有关说明，请参见。

步骤

1. 在要使用Asta Control进行管理的集群上安装Asta Connector运算符。运行此命令时、命名空间 `astra-connector-operator` 创建并将配置应用于命名空间：

```
kubectl apply -f https://github.com/NetApp/astra-connector-operator/releases/download/24.02.0-202403151353/astraconnector_operator.yaml
```

2. 确认操作员已安装并准备就绪：

```
kubectl get all -n astra-connector-operator
```

3. 从Asta Control获取API令牌。请参见 "Astra Automation文档" 有关说明，请参见。
4. 使用令牌创建密钥。将<API_TOKEN>替换为您从Astra Control收到的令牌：

```
kubectl create secret generic astra-token \
--from-literal=apiToken=<API_TOKEN> \
-n astra-connector
```

5. 创建Docker密钥以提取Astra Connector映像。将括号<>中的值替换为您环境中的信息：

> 您可以在Astra Control Web UI中找到<ASTRA_CONTROL_ACCOUNT_ID>。在Web UI中，选择页面右上角的图图标，然后选择*API access*。

```
kubectl create secret docker-registry regcred \
--docker-username=<ASTRA_CONTROL_ACCOUNT_ID> \
--docker-password=<API_TOKEN> \
-n astra-connector \
--docker-server=cr.astra.netapp.io
```

6. 创建Astra Connector CR文件并将其命名为 `astra-connector-cr.yaml`。更新方括号<>中的值以匹配您的Astra Control环境和集群配置：
   - <ASTRA_CONTROL_ACCOUNT_ID>：在上一步中从Astra Control Web UI获取。
   - <CLUSTER_NAME>：应在Asta Control中分配此集群的名称。
   - <ASTRA_CONTROL_URL>：Asta Control的Web UI URL。例如：

```
https://astra.control.url
```

```
apiVersion: astra.netapp.io/v1
kind: AstraConnector
metadata:
  name: astra-connector
  namespace: astra-connector
spec:
  astra:
    accountId: <ASTRA_CONTROL_ACCOUNT_ID>
    clusterName: <CLUSTER_NAME>
    #Only set `skipTLSValidation` to `true` when using the default
self-signed
    #certificate in a proof-of-concept environment.
    skipTLSValidation: false #Should be set to false in production
environments
    tokenRef: astra-token
  natsSyncClient:
    cloudBridgeURL: <ASTRA_CONTROL_HOST_URL>
  imageRegistry:
    name: cr.astra.netapp.io
    secret: regcred
```

7. 在您填充之后 `astra-connector-cr.yaml` 使用正确值的文件、应用CR：

```
kubectl apply -n astra-connector -f astra-connector-cr.yaml
```

8. 验证Asta Connector是否已完全部署：

```
kubectl get all -n astra-connector
```

9. 验证集群是否已注册到Astra Control：

```
kubectl get astraconnectors.astra.netapp.io -A
```

您应看到类似于以下内容的输出：

```
NAMESPACE          NAME               REGISTERED   ASTRACONNECTORID
STATUS
astra-connector    astra-connector    true         00ac8-2cef-41ac-8777-
ed0583e    Registered with Astra
```

10. 验证该集群是否显示在Astra Control Web UI的*集群*页面上的受管集群列表中。

添加集群

要开始管理应用程序，请添加 Kubernetes 集群并将其作为计算资源进行管理。您必须为 Astra 控制中心添加一个集群，才能发现您的 Kubernetes 应用程序。

我们建议，在将其他集群添加到 Astra 控制中心进行管理之前，先由 Astra 控制中心管理其部署所在的集群。要发送 Kubemmetrics 数据和集群关联数据以获取指标和故障排除信息，必须对初始集群进行管理。

开始之前

- 在添加集群之前，请查看并执行必要的操作 "前提条件任务"。
- 如果您使用的是ONTAP SAN驱动程序、请确保在所有Kubbernetes集群上启用了多路径。

步骤

1. 从信息板或集群菜单导航：
   - 从"Resource Summary"的"信息板"中、从"Clusters"窗格中选择"添加"。
   - 在左侧导航区域中、选择*集群*、然后从集群页面中选择*添加集群*。
2. 在打开的 * 添加集群 * 窗口中，上传 `kubeconfig.yaml` 文件或粘贴 `kubeconfig.yaml` 文件的内容。

   `kubeconfig.yaml` 文件应仅包含一个集群的集群凭据 * 。

   创建自己的 `kubeconfig` file中、您只能定义*一*上下文元素。请参见 "Kubernetes 文档" 有关创建的信息 `kubeconfig` 文件。如果您使用为有限集群角色创建了kubeconfig "此过程"、请务必在此步骤中上传或粘贴kubeconfig。
3. 请提供凭据名称。默认情况下，凭据名称会自动填充为集群的名称。
4. 选择 * 下一步 * 。
5. 选择要用于此Kubernetes集群的默认存储类、然后选择*下一步*。

   您应选择一个存储类、该存储类在Astra控件配置程序中进行配置、并由ONTAP存储提供支持。
6. 查看相关信息、如果一切正常、请选择*添加*。

结果

集群将进入*正在发现*状态、然后更改为*运行状况良好*。现在、您正在使用Astra控制中心管理集群。

添加要在 Astra 控制中心中管理的集群后，部署监控操作员可能需要几分钟的时间。在此之前，通知图标将变为红色并记录一个 * 监控代理状态检查失败 * 事件。您可以忽略此问题，因为当 Astra 控制中心获得正确状态时，问题描述将解析。如果问题描述 在几分钟内未解析、请转至集群并运行 `oc get pods -n netapp-monitoring` 作为起点。您需要查看监控操作员日志以调试问题。

在ONTAP存储后端启用身份验证

Astra控制中心提供了两种对ONTAP 后端进行身份验证的模式：

- 基于凭据的身份验证：具有所需权限的ONTAP 用户的用户名和密码。您应使用预定义的安全登录角色(如admin或vsadmin)、以确保与ONTAP 版本的最大兼容性。
- 基于证书的身份验证：Astra控制中心还可以使用后端安装的证书与ONTAP 集群进行通信。您应使用客户端证书、密钥和可信CA证书(如果使用)(建议)。

您可以稍后更新现有后端、以便从一种身份验证类型迁移到另一种身份验证方法。一次仅支持一种身份验证方法。

启用基于凭据的身份验证

ASRA控制中心需要集群范围的凭据 `admin` 与ONTAP 后端通信。您应使用标准的预定义角色、例如 `admin`。这样可以确保与未来的ONTAP 版本向前兼容、这些版本可能会公开功能API、以供未来的Astra控制中心版本使用。

> 可以创建自定义安全登录角色并将其用于Astra Control Center、但不建议这样做。

示例后端定义如下所示：

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

后端定义是以纯文本格式存储凭据的唯一位置。创建或更新后端是唯一需要了解凭据的步骤。因此、这是一项仅由管理员执行的操作、由Kubornetes或存储管理员执行。

启用基于证书的身份验证

Astra控制中心可以使用证书与新的和现有的ONTAP 后端进行通信。您应在后端定义中输入以下信息。

- `clientCertificate`：客户端证书。
- `clientPrivateKey`:关联的私钥。
- `trustedCACertificate`：可信CA证书。如果使用可信 CA ，则必须提供此参数。如果不使用可信 CA ，则可以忽略此设置。

您可以使用以下类型的证书之一：

- 自签名证书
- 第三方证书

使用自签名证书启用身份验证

典型的工作流包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时、请将公用名(Common Name、CN)设置为ONTAP 用户、以进行身份验证。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. 安装类型为的客户端证书 `client-ca` 和键ONTAP。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. 确认ONTAP 安全登录角色支持证书身份验证方法。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. 使用生成的证书测试身份验证。将<SVM ManagementLIF> and <vserver name> 替换为管理LIF IP和ONTAP 名称。您必须确保LIF的服务策略设置为 `default-data-management`。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. 使用上一步中获得的值、在Astra Control Center UI中添加存储后端。

使用第三方证书启用身份验证

如果您拥有第三方证书、则可以使用以下步骤设置基于证书的身份验证。

步骤

1. 生成私钥和CSR：

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. 将CSR传递到Windows CA (第三方CA)、然后问题描述 签名证书。
3. 下载签名证书并将其命名为`ONTAP signed_cert.crt
4. 从Windows CA (第三方CA)导出根证书。
5. 为此文件命名 `ca_root.crt`

   现在、您已有以下三个文件：

   - 私钥： `ontap_signed_request.key` (这是ONTAP 中服务器证书对应的密钥。安装服务器证书时需要此证书。)
   - 签名证书： `ontap_signed_cert.crt` (在ONTAP 中也称为_server certificATE _。)
   - 根**CA**证书： `ca_root.crt` (在ONTAP 中也称为_server-ca certific存在_。)
6. 在ONTAP 中安装这些证书。生成并安装 `server` 和 `server-ca` ONTAP 上的证书。

展开**SAMPLE.YAML**

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. 为同一主机创建客户端证书、以实现无密码通信。Asta控制中心使用此过程与ONTAP 进行通信。
8. 在ONTAP 上生成并安装客户端证书：

展开**SAMPLE.YAML**

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate's CA and serial number for reference:

CA:
serial:
The certificate's generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```

9. 在Asta Control Center UI中添加存储后端、并提供以下值：
   - 客户端证书：ONATP_TEST_client.prom
   - 私钥：ontap_test_client.key
   - 可信**CA**证书：ONATP_signed_cert.crt

添加存储后端

设置凭据或证书身份验证信息后、您可以将现有ONTAP 存储后端添加到Astra控制中心以管理其资源。

通过将 Astra Control 中的存储集群作为存储后端进行管理，您可以在永久性卷（ PV ）和存储后端之间建立链接，并获得其他存储指标。

如果启用了Astra Control配置程序、则在使用NetApp SnapMirror技术时、可以选择在Astra控制中心中添加和管理ONTAP存储后端。

步骤

1. 从左侧导航区域的信息板中、选择*后端*。
2. 选择 * 添加 * 。
3. 在添加存储后端页面的使用现有部分中，选择* ONTAP *。
4. 选择以下选项之一：
   - 使用管理员凭据：输入ONTAP 集群管理IP地址和管理员凭据。凭据必须是集群范围的凭据。

     您在此处输入凭据的用户必须具有 `ontapi` 在ONTAP 集群上的ONTAP 系统管理器中启用用户登录访问方法。如果您计划使用SnapMirror复制、请应用具有"admin"角色的用户凭据、该角色具有访问方法 `ontapi` 和 `http`、在源和目标ONTAP 集群上。请参见 "管理ONTAP 文档中的用户帐户" 有关详细信息 ...

   - 使用证书：上传证书 `.pem` file、证书密钥 `.key` 文件、以及证书颁发机构文件(可选)。
5. 选择 * 下一步 * 。
6. 确认后端详细信息并选择 * 管理 * 。

结果

后端将显示在中 `online` 包含摘要信息的列表中的状态。

您可能需要刷新页面才能显示后端。

添加存储分段

您可以使用Astra Control UI或添加存储分段 "Astra Control API"。如果要备份应用程序和永久性存储，或者要跨集群克隆应用程序，则必须添加对象存储分段提供程序。Astra Control 会将这些备份或克隆存储在您定义的对象存储分段中。

如果您要将应用程序配置和永久性存储克隆到同一集群、则无需在Astra Control中使用存储分段。应用程序快照功能不需要存储分段。

开始之前

- 确保您有一个可从Astra Control Center管理的集群访问的存储分段。
- 确保您具有此存储分段的凭据。
- 确保存储分段为以下类型之一：
  - NetApp ONTAP S3
  - NetApp StorageGRID S3
  - Microsoft Azure
  - 通用 S3

Amazon Web Services (AWS)和Google Cloud Platform (GCP)使用通用S3存储分段类型。

虽然Astra控制中心支持将Amazon S3作为通用S3存储分段提供商、但Astra控制中心可能不支持声称支持Amazon S3的所有对象存储供应商。

步骤

1. 在左侧导航区域中，选择 * 桶 * 。
2. 选择 * 添加 * 。
3. 选择存储分段类型。

   添加存储分段时，请选择正确的存储分段提供程序，并为该提供程序提供正确的凭据。例如， UI 接受 NetApp ONTAP S3 作为类型并接受 StorageGRID 凭据；但是，这将发生原因使使用此存储分段执行所有未来应用程序备份和还原失败。

4. 输入现有存储分段名称和可选的问题描述。

   存储分段名称和问题描述 显示为备份位置、您可以稍后在创建备份时选择该位置。此名称也会在配置保护策略期间显示。

5. 输入 S3 端点的名称或 IP 地址。
6. 在*选择凭据*下、选择*添加*或*使用现有*选项卡。
   - 如果选择*添加*：
     i. 在 Astra Control 中输入凭据名称，以便与其他凭据区分开。
     ii. 通过粘贴剪贴板中的内容来输入访问 ID 和机密密钥。
   - 如果选择*使用现有*：
     i. 选择要用于存储分段的现有凭据。
7. 选择 … `Add`。

   添加存储分段时、Astra Control会使用默认存储分段指示符标记一个存储分段。您创建的第一个存储分段将成为默认存储分段。添加分段时、您可以稍后决定添加 "设置另一个默认存储分段"。