安装概述

Astra Control Center

NetApp
August 11, 2025

目录

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控制中心、请从NetApp 支持站点 下载安装包并执行以下步骤。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。

展开以了解其他安装过程

- 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。

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。
- 访问**NetApp Astra**控件映像注册表：
  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。

展开步骤

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>
```

- 考虑服务网格：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"。

Isio服务网格详细信息

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
```

- 仅限**ONTAP SAN**驱动程序：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。

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
- 安装NetApp Astra kubectl插件
- [将映像添加到本地注册表]
- [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

您可以选择从NetApp 支持站点 下载Astra控制中心包、也可以使用Docker从Astra控制服务映像注册表中提取该包。

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。

   展开以查看详细信息

   ```
   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
   ```

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。

安装NetApp Astra kubectl插件

您可以使用NetApp Asta kubect命令行插件将映像推送到本地Docker存储库。

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
   ```

将映像添加到本地注册表

1. 为容器引擎完成相应的步骤顺序：

Docker

1. 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
```

2. 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

1. 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
```

2. 登录到注册表：

```
podman login <YOUR_REGISTRY>
```

3. 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=23.10.0-68
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=23.10.0-68
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/23.10.0-68/image:version
```

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：

展开步骤

a. 创建 `netapp-acc-operator` 命名空间：

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

b. 为创建密钥 `netapp-acc-operator` 命名空间。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/23.10.0-68`）。

```
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。

c. 创建 `netapp-acc` (或自定义命名的)命名空间。

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

d. 为创建密钥 `netapp-acc` (或自定义命名的)命名空间。添加 Docker 信息并运行以下命令：

```
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. 更改目录：

```
cd manifests
```

2. 编辑Astra控制中心操作员部署YAML (`astra_control_center_operator_deploy.yaml`)以引用您的本地注册表和密钥。

```
vim astra_control_center_operator_deploy.yaml
```

以下步骤将提供一个标注的YAML示例。

a. 如果您使用的注册表需要身份验证、请替换的默认行 `imagePullSecrets: []` 使用以下命令：

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

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

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

展开以获取示例**Astra_control_cCenter_operator_Deploy。yaml**

```
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:23.10.72
        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
```

3. 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
```

4. 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：

<code>accountName</code>

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`

<code>astraVersion</code>

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	`23.10.0-68`

<code>astraAddress</code>

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`

<code>autoSupport</code>

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到NetApp以获得支持。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`

<code>email</code>

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`

<code>firstName</code>

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`

<code>LastName</code>

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`

<code>imageRegistry</code>

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	在中推送映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	`example.registry.com/astra`
`imageRegistry.secret`	如果您为输入的字符串、则为必填项 `imageRegistry.name' requires a secret.` `IMPORTANT: If you are using a registry that does not require authorization, you must delete this` `secret` 行内 `imageRegistry` 否则安装将失败。	用于通过映像注册表进行身份验证的Kubernetes密钥的名称。	string	`astra-registry-cred`

<code>storageClass</code>

正在设置 ...	指导	Type	示例
`storageClass`	更改 `storageClass` 价值来自 `ontap-gold` 另一个A作用 于安装所需的Astra三端存储类资源。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一个基于Astra三端的存储类 (`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`

<code>volumeReclaimPolicy</code>

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`

<code>ingressType</code>

正在设置 ...	指导	Type	选项
`ingressType`	请使用以下入口类型之一： `Generic`* (`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控制中心后、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`

<code>scaleSize</code>

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)

<code>astraResourcesScaler</code>

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`

<code>additionalValues</code>

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
```

- 对于Astral控制中心和Cloud Insights 通信、默认情况下会禁用TLS证书验证。您可以通过在中添加以下部分来为Cloud Insights 与Astra控制中心主机集群和受管集群之间的通信启用TLS证书验证 `additionalValues`。

```
 additionalValues:
   netapp-monitoring-operator:
     config:
       ciSkipTlsVerify: false
   cloud-insights-service:
     config:
       ciSkipTlsVerify: false
   telemetry-service:
     config:
       ciSkipTlsVerify: false
```

<code>crds</code>

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。

展开示例**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: "[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执行此操作、请创建 `netapp-acc` (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. 在中安装Astra控制中心 `netapp-acc` (或自定义)命名空间：

```
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

```
NAME                                          READY   STATUS
RESTARTS     AGE
acc-helm-repo-6cc7696d8f-pmhm8                1/1     Running     0
9h
activity-597fb656dc-5rd4l                     1/1     Running     0
9h
activity-597fb656dc-mqmcw                     1/1     Running     0
9h
api-token-authentication-62f84                1/1     Running     0
9h
api-token-authentication-68nlf                1/1     Running     0
9h
api-token-authentication-ztgrm                1/1     Running     0
9h
asup-669d4ddbc4-fnmwp                         1/1     Running     1
(9h ago)   9h
authentication-78789d7549-lk686               1/1     Running     0
9h
bucketservice-65c7d95496-24x7l                1/1     Running     3
(9h ago)   9h
cert-manager-c9f9fbf9f-k8zq2                  1/1     Running     0
9h
cert-manager-c9f9fbf9f-qjlzm                  1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-b5qll       1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-p5whs       1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-4722b         1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-86kv5         1/1     Running     0
9h
certificates-59d9f6f4bd-2j899                 1/1     Running     0
9h
certificates-59d9f6f4bd-9d9k6                 1/1     Running     0
9h
certificates-expiry-check-28011180--1-8lkxz   0/1     Completed   0
9h
cloud-extension-5c9c9958f8-jdhrp              1/1     Running     0
9h
cloud-insights-service-5cdd5f7f-pp8r5         1/1     Running     0
9h
composite-compute-66585789f4-hxn5w            1/1     Running     0
```

```
9h
composite-volume-68649f68fd-tb7p4             1/1     Running     0
9h
credentials-dfc844c57-jsx92                   1/1     Running     0
9h
credentials-dfc844c57-xw26s                   1/1     Running     0
9h
entitlement-7b47769b87-4jb6c                  1/1     Running     0
9h
features-854d8444cc-c24b7                     1/1     Running     0
9h
features-854d8444cc-dv6sm                     1/1     Running     0
9h
fluent-bit-ds-9tlv4                           1/1     Running     0
9h
fluent-bit-ds-bpkcb                           1/1     Running     0
9h
fluent-bit-ds-cxmwx                           1/1     Running     0
9h
fluent-bit-ds-jgnhc                           1/1     Running     0
9h
fluent-bit-ds-vtr6k                           1/1     Running     0
9h
fluent-bit-ds-vxqd5                           1/1     Running     0
9h
graphql-server-7d4b9d44d5-zdbf5               1/1     Running     0
9h
identity-6655c48769-4pwk8                     1/1     Running     0
9h
influxdb2-0                                   1/1     Running     0
9h
keycloak-operator-55479d6fc6-slvmt            1/1     Running     0
9h
krakend-f487cb465-78679                       1/1     Running     0
9h
krakend-f487cb465-rjsxx                       1/1     Running     0
9h
license-64cbc7cd9c-qxsr8                      1/1     Running     0
9h
login-ui-5db89b5589-ndb96                     1/1     Running     0
9h
loki-0                                        1/1     Running     0
9h
metrics-facade-8446f64c94-x8h7b               1/1     Running     0
9h
monitoring-operator-6b44586965-pvcl4          2/2     Running     0
```

```
9h
nats-0                                            1/1     Running     0
9h
nats-1                                            1/1     Running     0
9h
nats-2                                            1/1     Running     0
9h
nautilus-85754d87d7-756qb                         1/1     Running     0
9h
nautilus-85754d87d7-q8j7d                         1/1     Running     0
9h
openapi-5f9cc76544-7fnjm                          1/1     Running     0
9h
openapi-5f9cc76544-vzr7b                          1/1     Running     0
9h
packages-5db49f8b5-lrzhd                          1/1     Running     0
9h
polaris-consul-consul-server-0                    1/1     Running     0
9h
polaris-consul-consul-server-1                    1/1     Running     0
9h
polaris-consul-consul-server-2                    1/1     Running     0
9h
polaris-keycloak-0                                1/1     Running     2
(9h ago)   9h
polaris-keycloak-1                                1/1     Running     0
9h
polaris-keycloak-2                                1/1     Running     0
9h
polaris-keycloak-db-0                             1/1     Running     0
9h
polaris-keycloak-db-1                             1/1     Running     0
9h
polaris-keycloak-db-2                             1/1     Running     0
9h
polaris-mongodb-0                                 1/1     Running     0
9h
polaris-mongodb-1                                 1/1     Running     0
9h
polaris-mongodb-2                                 1/1     Running     0
9h
polaris-ui-66fb99479-qp9gq                        1/1     Running     0
9h
polaris-vault-0                                   1/1     Running     0
9h
polaris-vault-1                                   1/1     Running     0
```

```
9h
polaris-vault-2                                   1/1     Running     0
9h
public-metrics-76fbf9594d-zmxzw                   1/1     Running     0
9h
storage-backend-metrics-7d7fbc9cb9-lmd25          1/1     Running     0
9h
storage-provider-5bdd456c4b-2fftc                 1/1     Running     0
9h
task-service-87575df85-dnn2q                      1/1     Running     3
(9h ago)   9h
task-service-task-purge-28011720--1-q6w4r         0/1     Completed   0
28m
task-service-task-purge-28011735--1-vk6pd         1/1     Running     0
13m
telegraf-ds-2r2kw                                 1/1     Running     0
9h
telegraf-ds-6s9d5                                 1/1     Running     0
9h
telegraf-ds-96jl7                                 1/1     Running     0
9h
telegraf-ds-hbp84                                 1/1     Running     0
9h
telegraf-ds-plwzv                                 1/1     Running     0
9h
telegraf-ds-sr22c                                 1/1     Running     0
9h
telegraf-rs-4sbg8                                 1/1     Running     0
9h
telemetry-service-fb9559f7b-mk9l7                 1/1     Running     3
(9h ago)   9h
tenancy-559bbc6b48-5msgg                          1/1     Running     0
9h
traefik-d997b8877-7xpf4                           1/1     Running     0
9h
traefik-d997b8877-9xv96                           1/1     Running     0
9h
trident-svc-585c97548c-d25z5                      1/1     Running     0
9h
vault-controller-88484b454-2d6sr                  1/1     Running     0
9h
vault-controller-88484b454-fc5cz                  1/1     Running     0
9h
vault-controller-88484b454-jktld                  1/1     Running     0
9h
```

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`)并获取登录到Astra控制中心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  23.10.0-68
10.111.111.111  True
```

> 复制UUID值。密码为 `ACC-` 后跟UUID值 (`ACC-[UUID]` 或者、在此示例中、 `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 控制中心后，您需要配置入口控制器，以便使用 URL 公开 Astra 控制中心。

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
   ```

   密钥名称应与匹配 `spec.tls.secretName` 在中提供 `istio-ingress.yaml` 文件

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 控制中心后，您将更改默认管理员的密码并登录到 Astra 控制中心 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

如果有任何服务位于中 `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
```

下一步行动

- (可选)根据您的环境、完成安装后操作 "配置步骤"。
- 执行以完成部署 "设置任务"。

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"。
- 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。
- 考虑服务网格：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"。

Isio服务网格详细信息

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
```

- * Kubernetes入口控制器*：如果您的Kubernetes入口控制器负责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控制中心配合使用：

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：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。

步骤

- 下载并提取Astra控制中心
- 安装NetApp Astra kubectl插件
- [将映像添加到本地注册表]
- [找到操作员安装页面]
- [安装操作员]
- 安装 Astra 控制中心

下载并提取Astra控制中心

您可以选择从NetApp 支持站点 下载Astra控制中心包、也可以使用Docker从Astra控制服务映像注册表中提取该包。

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。

   展开以查看详细信息

   ```
   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
   ```

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。

安装NetApp Astra kubectl插件

您可以使用NetApp Asta kubect命令行插件将映像推送到本地Docker存储库。

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。

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
   ```

将映像添加到本地注册表

1. 为容器引擎完成相应的步骤顺序：

Docker

1. 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
```

2. 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

1. 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
```

2. 登录到注册表：

```
podman login <YOUR_REGISTRY>
```

3. 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=23.10.0-68
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=23.10.0-68
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/23.10.0-68/image:version
```

找到操作员安装页面

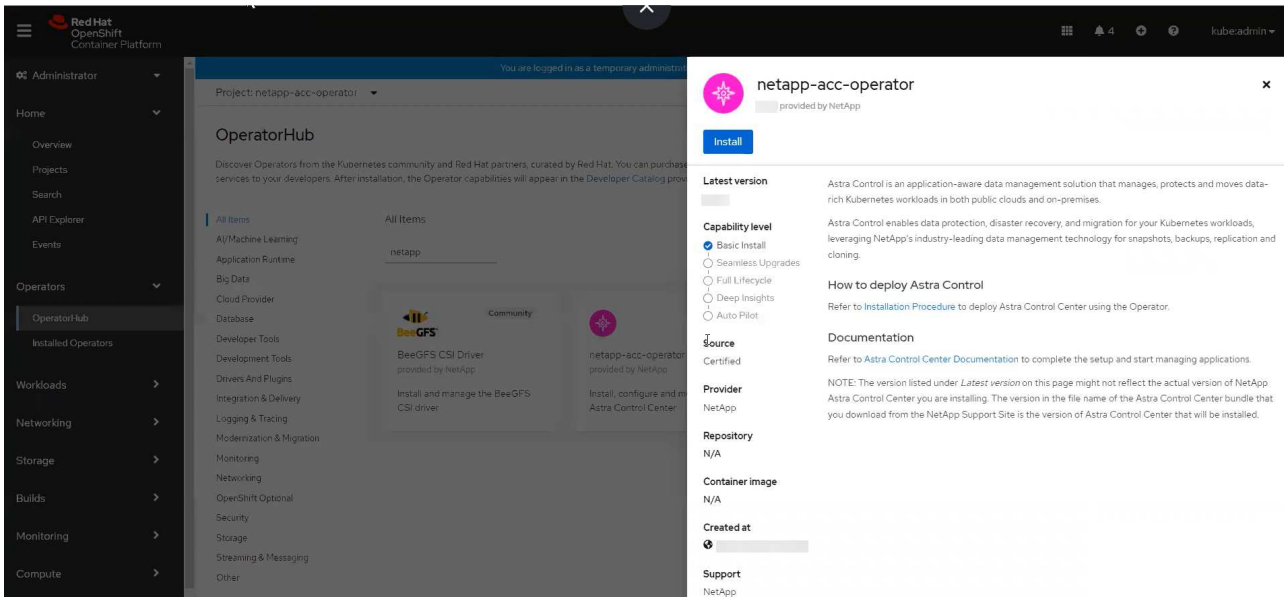
1. 要访问操作员安装页面，请完成以下过程之一：
   - 从Red Hat OpenShift Web控制台：

i. 登录到 OpenShift 容器平台 UI 。

ii. 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。

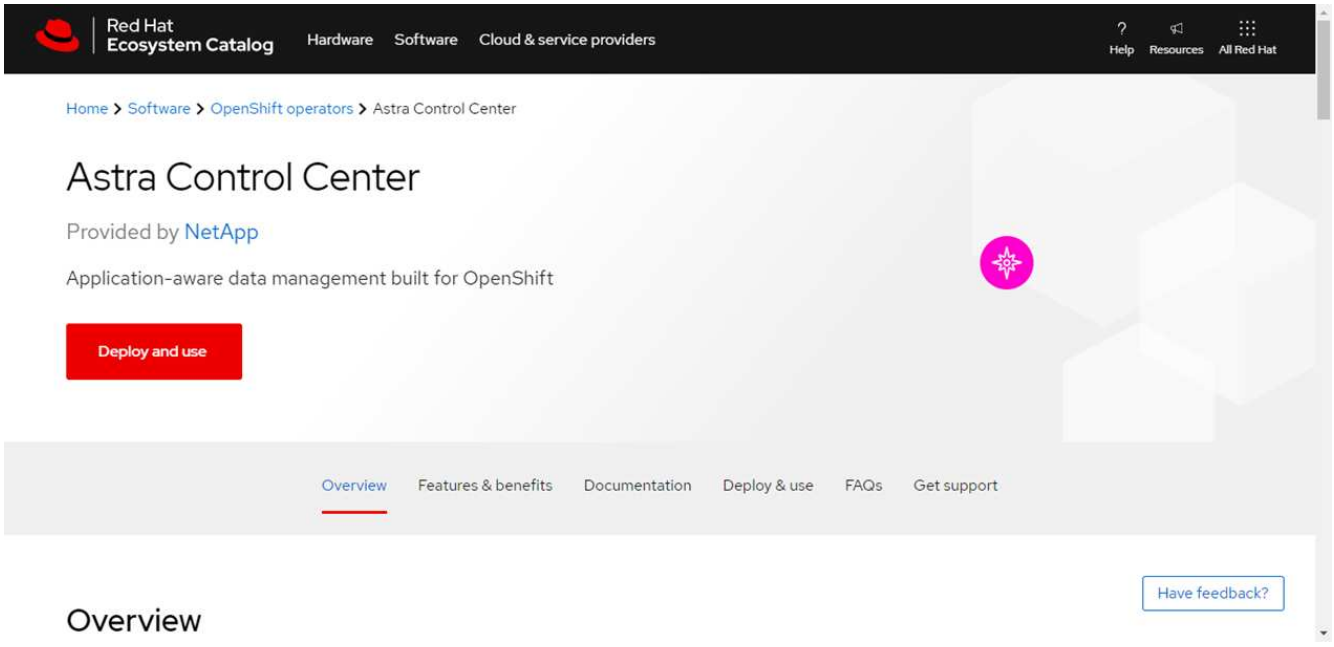
iii. 搜索并选择NetApp Astra Control Center运算符。



◦ 从 Red Hat 生态系统目录：

i. 选择 NetApp Astra 控制中心 "运算符"。

ii. 选择 * 部署并使用 * 。



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. 选择操作符命名空间或 `netapp-acc-operator` 命名空间将在操作员安装过程中自动创建。

   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。

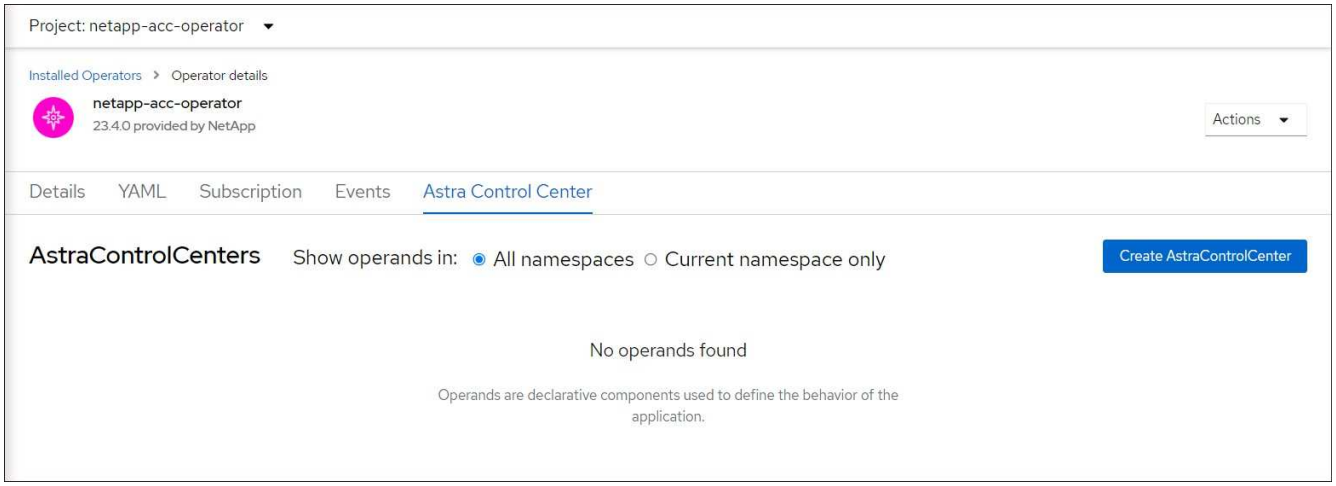
   c. 选择 * 安装 * 。

      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. 完成 `Create AstraControlCenter` 表单字段：

   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23.10.0-68。

   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. 选择安装的可扩展大小。

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：

- **`Generic`** (`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控制中心后、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 * 映像注册表 * 中，输入本地容器映像注册表路径。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 Kubernetes 集群或云环境中的一个自管理 Kubernetes 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 控制中心之前，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， Astra 控制中心还需要以下资源：

组件	要求
后端 **NetApp Cloud Volumes ONTAP** 存储容量	至少 300 GB 可用
工作节点（**AWS EC2** 要求）	总共至少 3 个辅助节点，每个节点有 4 个 vCPU 核心和 12 GB RAM
负载平衡器	服务类型 "loadbalancer" 可用于将传入流量发送到操作环境集群中的服务
FQDN	一种将 Astra 控制中心的 FQDN 指向负载平衡 IP 地址的方法
Astra Trident (在**NetApp BlueXP** 中作为**Kubernetes**集群发现的一部分安装、以前称为**Cloud Manager)**	已安装并配置Asta Trident 23.01或更高版本、并将NetApp ONTAP 9.9.1或更高版本用作存储后端[AWS注册表]]
映像注册表	NetApp提供了一个注册表、可用于获取Astra控制中心内部版本映像：http://netappdownloads.jfrog.io/docker-astra-control-prod 请联系NetApp支持部门、获取有关在Astra控制中心安装过程中使用此映像注册表的说明。 如果您无法访问NetApp映像注册表、则必须具有现有的私有注册表、例如AWS Elastic Container Registry (ECR)、您可以将Astra控制中心构建映像推送到该注册表。您需要提供要将映像上传到的映像注册表的 URL 。 Astra 控制中心托管的集群和受管集群必须能够访问同一映像注册表，才能使用基于 Restic 的映像备份和还原应用程序。

组件	要求
Astra Trident / ONTAP 配置	Astra 控制中心要求创建一个存储类并将其设置为默认存储类。Astra控制中心支持以下ONTAP Kubernetes存储类、这些存储类是在将Kubernetes集群导入到NetApp BlueXP (以前称为Cloud Manager)时创建的。这些功能由 Astra Trident 提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

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。

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 Asta控制中心映像注册表，您还需要创建一个Elastic Container Registry (ECR)来托管Astra Control Center映像，并将这些映像推送到该注册表。

按照 AWS 文档完成以下步骤。请参见 "AWS 安装文档"。

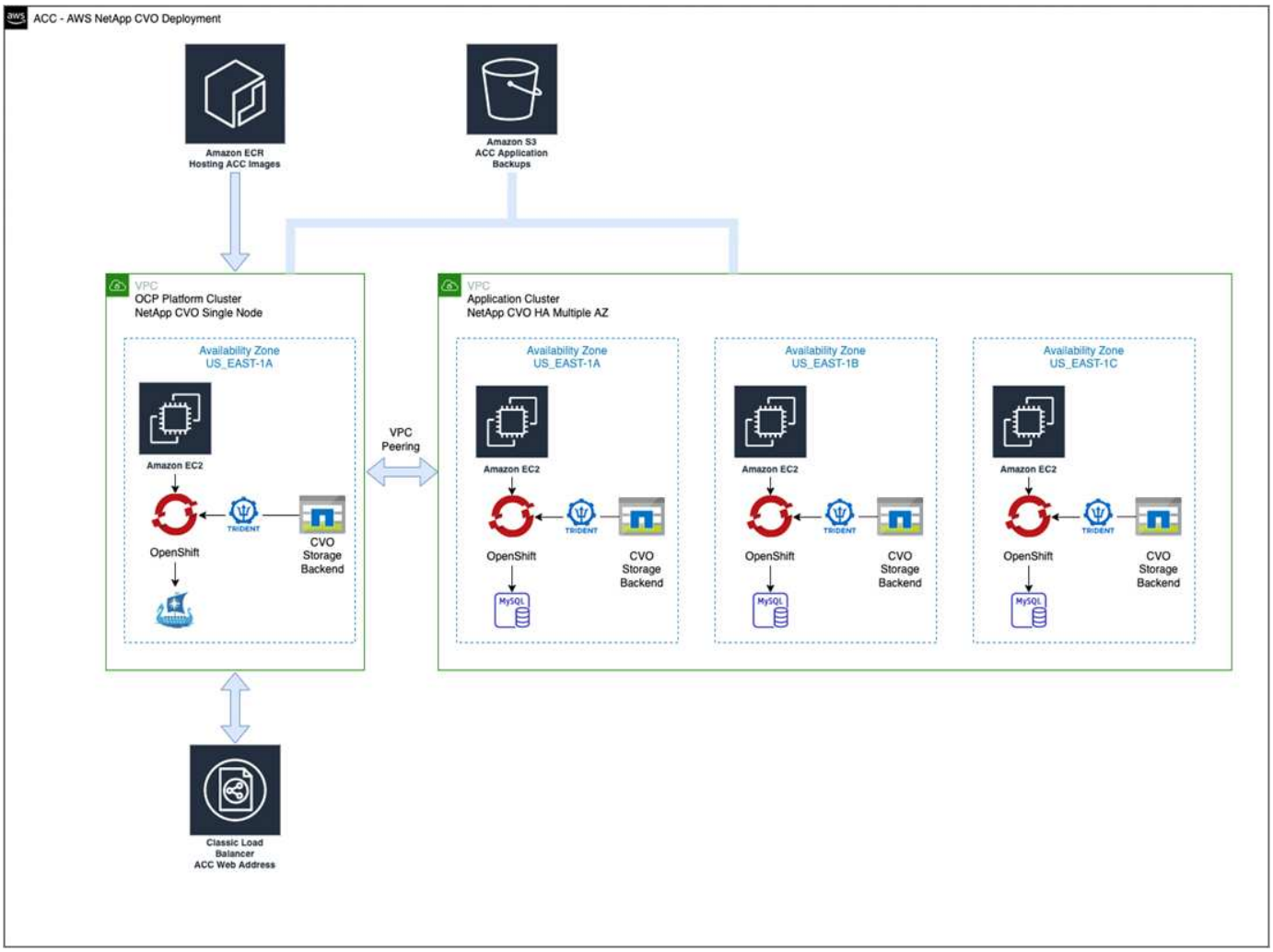
1. 创建AWS虚拟网络。
2. 查看 EC2 计算实例。这可以是 AWS 中的裸机服务器或 VM 。
3. 如果实例类型尚未与主节点和工作节点的 Astra 最低资源要求匹配，请更改 AWS 中的实例类型以满足 Astra 要求。 请参见 "Astra 控制中心要求"。
4. 至少创建一个 AWS S3 存储分段来存储备份。
5. (可选)如果无法访问 NetApp映像注册表，请执行以下操作：
   a. 创建AWS Elastic Container Registry (ECR)以托管所有Astra Control Center映像。

      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三端版本。

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三项功能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控制中心之前、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)
- 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， Astra 控制中心还需要以下资源：

组件	要求
后端 **NetApp Cloud Volumes ONTAP** 存储容量	至少 300 GB 可用
工作节点**(GCP**计算要求**)**	总共至少 3 个辅助节点，每个节点有 4 个 vCPU 核心和 12 GB RAM
负载平衡器	服务类型 "loadbalancer" 可用于将传入流量发送到操作环境集群中的服务

组件	要求
FQDN (GCP DNS区域)	一种将 Astra 控制中心的 FQDN 指向负载平衡 IP 地址的方法
Astra Trident (在NetApp BlueXP中作为Kubernetes集群发现的一部分安装、以前称为Cloud Manager)	已安装并配置Astra Trident 23.01或更高版本、并将NetApp ONTAP 9.9.1或更高版本用作存储后端[gcp-Registry]
映像注册表	NetApp提供了一个注册表、可用于获取Astra控制中心内部版本映像：`http://netappdownloads.jfrog.io/docker-astra-control-prod` 请联系NetApp支持部门、获取有关在Astra控制中心安装过程中使用此映像注册表的说明。 如果您无法访问NetApp映像注册表、则必须具有现有的私有注册表、例如Google容器注册表、您可以将Astra控制中心构建映像推送到该注册表。您需要提供要将映像上传到的映像注册表的 URL 。 您需要启用匿名访问以提取要备份的 Restic 映像。
Astra Trident / ONTAP 配置	Astra 控制中心要求创建一个存储类并将其设置为默认存储类。Astra控制中心支持在将ONTAP Kubernetes集群导入到NetApp BlueXP中时创建的以下Kubernetes存储类。这些功能由 Astra Trident 提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

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。

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Asta Control Center。

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 Asta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 Control Center映像，并将这些映像推送到该注册表。

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 (原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. 在右上角，记下 Trident 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三项功能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**Asta Control Center**

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 控制中心之前，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， Astra 控制中心还需要以下资源：

请参见 "Astra 控制中心运营环境要求"。

组件	要求
后端 **NetApp Cloud Volumes ONTAP** 存储容量	至少 300 GB 可用
员工节点（**Azure** 计算要求）	总共至少 3 个辅助节点，每个节点有 4 个 vCPU 核心和 12 GB RAM
负载平衡器	服务类型 "loadbalancer" 可用于将传入流量发送到操作环境集群中的服务
FQDN （**Azure DNS** 区域）	一种将 Astra 控制中心的 FQDN 指向负载平衡 IP 地址的方法
Astra Trident (在**NetApp BlueXP** 中作为**Kubernetes**集群发现的一部分安装**)**	已安装并配置Asta Trident 23.01或更高版本、并且NetApp ONTAP 9.9.1或更高版本将用作存储后端[[azure-Registry]]
映像注册表	NetApp提供了一个注册表、可用于获取Astra控制中心内部版本映像：http://netappdownloads.jfrog.io/docker-astra-control-prod 请联系NetApp支持部门、获取有关在Astra控制中心安装过程中使用此映像注册表的说明。 如果您无法访问NetApp映像注册表、则必须具有一个现有的私有注册表、例如Azure容器注册表(ACR)、您可以将Astra控制中心构建映像推送到该注册表。您需要提供要将映像上传到的映像注册表的 URL 。 您需要启用匿名访问以提取要备份的 Restic 映像。
Astra Trident / ONTAP 配置	Astra 控制中心要求创建一个存储类并将其设置为默认存储类。Astra控制中心支持在将ONTAP Kubernetes集群导入到NetApp BlueXP中时创建的以下Kubernetes存储类。这些功能由 Astra Trident 提供： • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

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。

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。如果无法访问 NetApp Asta控制中心映像注册表，您还需要创建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。您需要一个存储帐户来创建要用作 Astra 控制中心分段的容器。
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

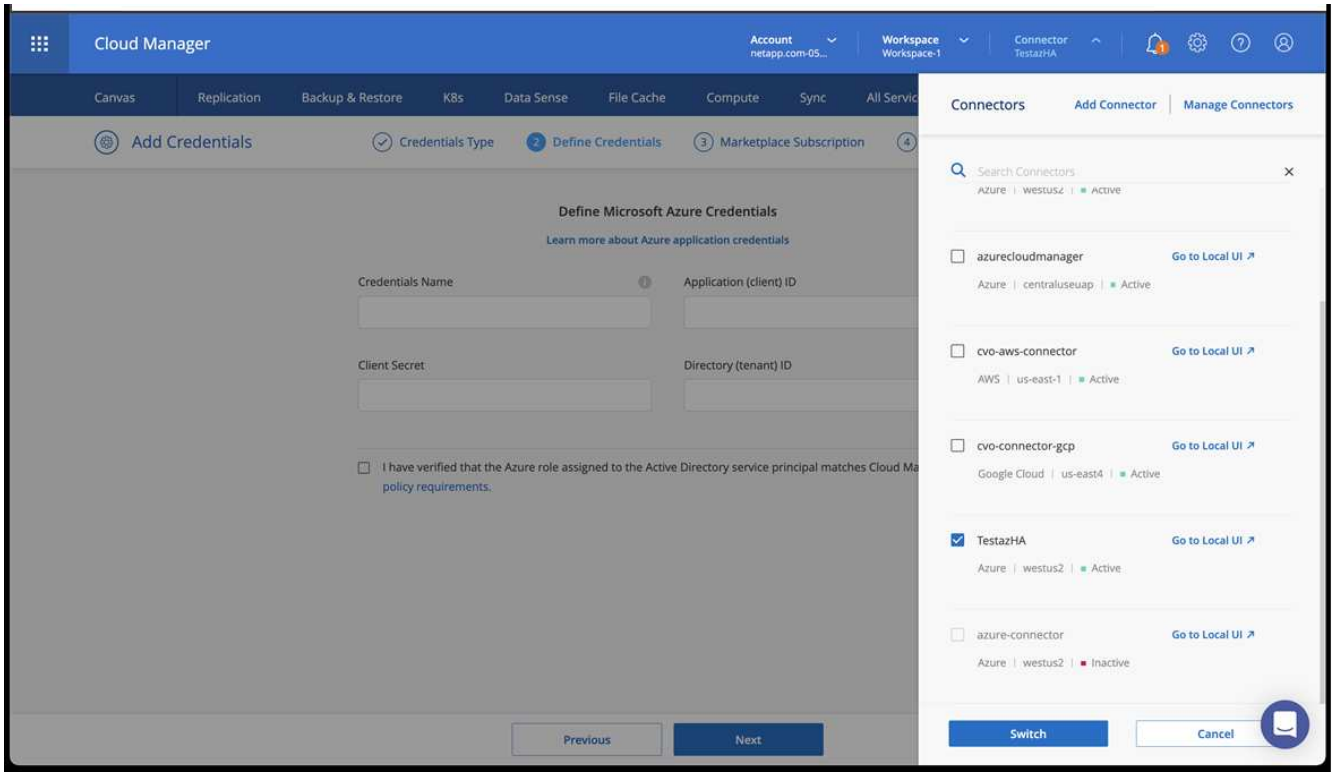
使用所需的 IAM 权限和角色访问 Azure 帐户

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"。

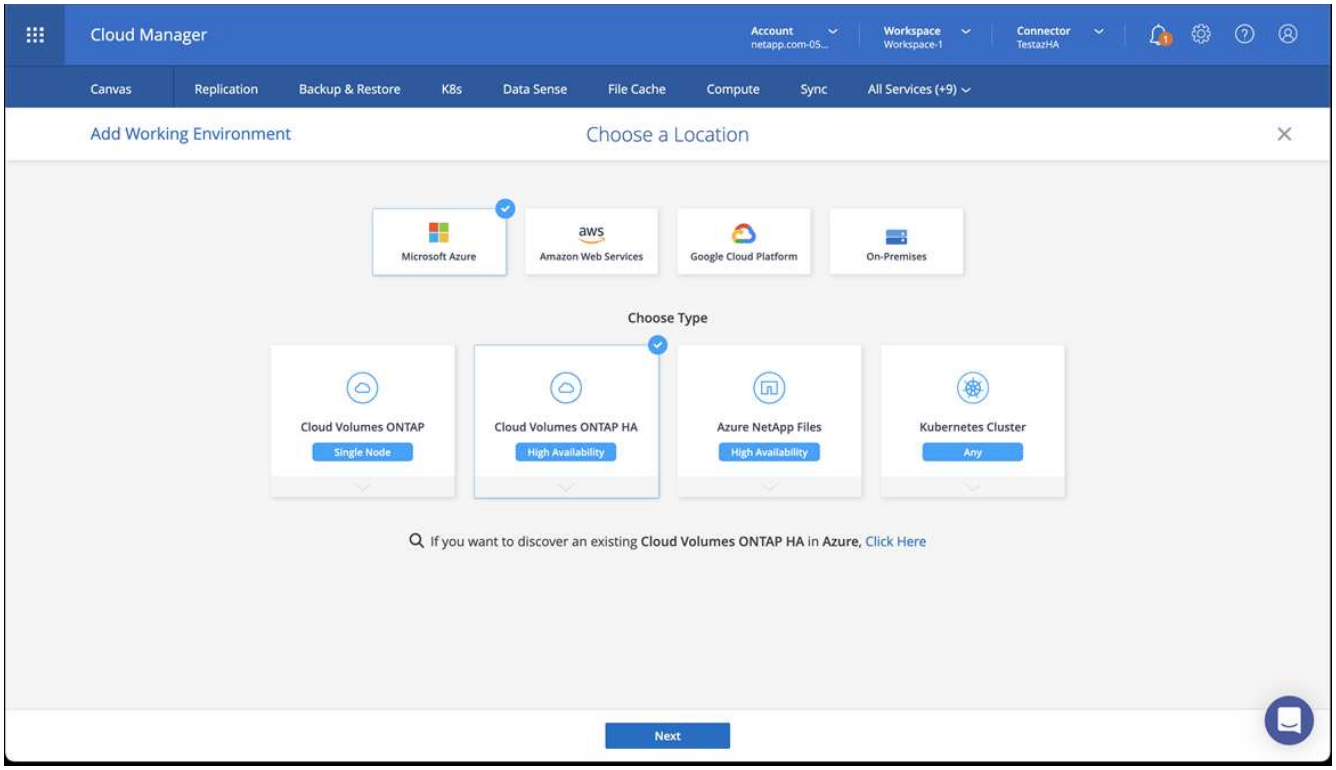
3. 确保连接器正在运行，然后切换到该连接器。



4. 为您的云环境创建一个工作环境。

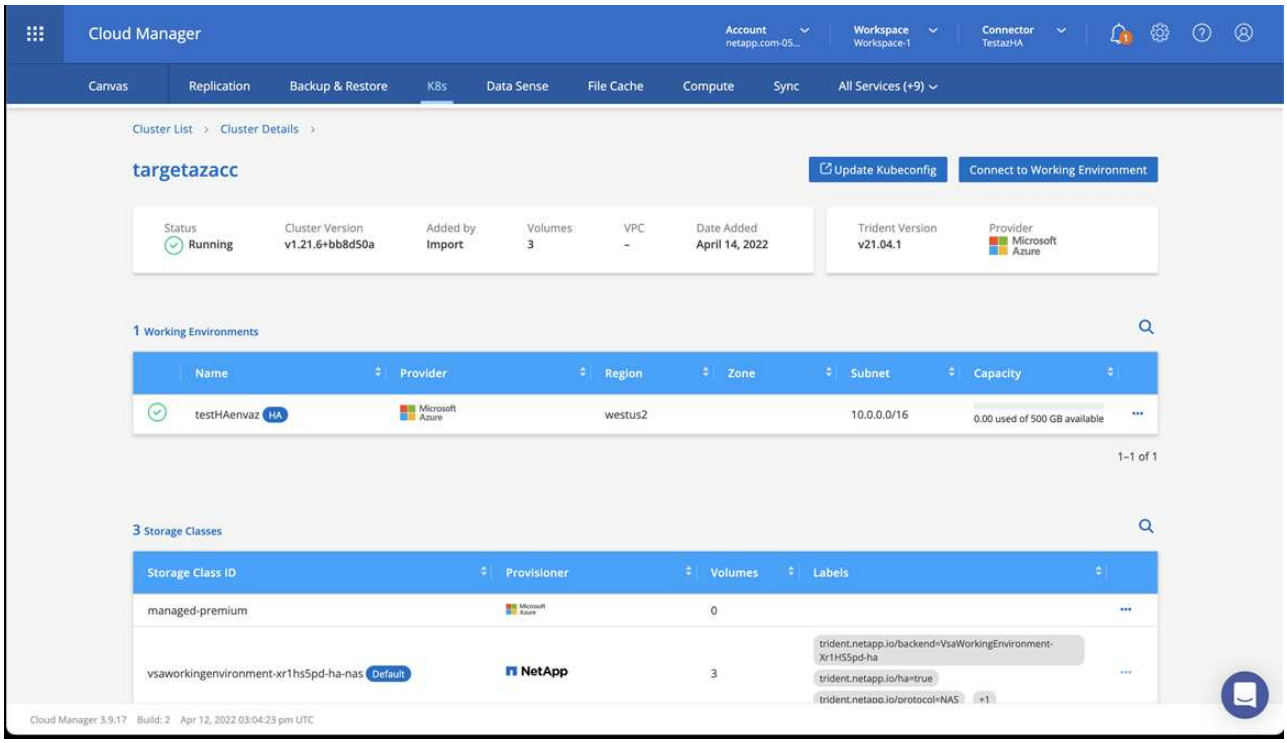
   a. 位置： "Microsoft Azure" 。

   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三端版本。

   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三项功能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。您可以删除现有的自签名 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
- 对集群上要运行的命令Shell的管理访问 `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. 使用以下命令替换、查找与当前证书关联的TLS密钥 `<ACC-deployment-namespace>` 使用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)文件并更改 `spec.selfSigned` 值为 `spec.ca.secretName` 要引用先前创建的TLS密钥、请执行以下操作：

```
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、然后进行替换 `<ACC-deployment-namespace>` 使用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. 创建 `certificate.yaml` file使用以下示例将括号<>中的占位符值替换为相应的信息：

```
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。