



安装 **Astra** 控制中心

Astra Control Center

NetApp
November 21, 2023

目录

使用标准流程安装 Astra 控制中心	1
下载并提取Astra控制中心	2
安装NetApp Astra kubectI插件	2
将映像添加到本地注册表	3
为具有身份验证要求的注册表设置命名空间和密钥	5
安装 Astra 控制中心操作员	7
配置 Astra 控制中心	10
完成 Astra 控制中心和操作员安装	19
验证系统状态	20
设置传入以进行负载平衡	25
登录到 Astra 控制中心 UI	29
对安装进行故障排除	29
下一步行动	30
配置外部证书管理器	30

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。

其他安装过程

- 使用**RedHat OpenShift OperatorHub**安装：使用此 "[备用操作步骤](#)" 使用OperatorHub在OpenShift上安装Astra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Astra控制中心安装过程的演示、请参见 "[此视频](#)"。

开始之前

- "[开始安装之前，请为 Astra Control Center 部署准备您的环境](#)"。
- 如果您已在环境中配置或希望配置POD安全策略、请熟悉POD安全策略及其对Astra Control Center安装的影响。请参见 "[了解POD安全策略限制](#)"。
- 确保所有 API 服务均处于运行状况良好且可用：

```
kubectl get apiservices
```

- 确保您计划使用的Astra FQDN可路由到此集群。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。
- 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。



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。

关于此任务

Astra控制中心安装过程可帮助您执行以下操作：

- 将Astra组件安装到中 netapp-acc (或自定义命名的)命名空间。
- 创建默认的Astra Control所有者管理员帐户。
- 建立管理用户电子邮件地址和默认初始设置密码。系统会为此用户分配首次登录到UI所需的所有者角色。
- 确定所有Astra控制中心Pod均正在运行。
- 安装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。

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](#)

下载并提取Astra控制中心

1. 转至 "[Astra Control Center 下载页面](#)" 页面。
2. 下载包含Astra Control Center的软件包 (`astra-control-center-[version].tar.gz`) 。
3. (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`)以验证捆绑包的签名：

```
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。

4. 从Astra Control Center捆绑包中提取映像：

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

安装NetApp Astra kubectl插件

您可以使用NetApp Astra kubectl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 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。您应看到此文件和目录：

```
acc.manifest.bundle.yaml
acc/
```

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.manifest.bundle.yaml
acc/
```

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=23.04.2-7
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
```

Podman 3

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.04.2-7
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://netappdownloads.jfrog.io/docker-astra-control-
prod/netapp/astra/acc/23.04.2-7/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控制中心主机集群的KUBECONFIG:

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



在完成安装之前、请确保您的KUBECONFIG指向要安装Astra控制中心的集群。KUBECONFIG只能包含一个上下文。

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
```

响应：

```
namespace/netapp-acc-operator created
```

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/23.04.2-7`)。

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

响应示例：

```
secret/astra-registry-cred created
```



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]
```

响应示例：

```
namespace/netapp-acc created
```


- 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]
```

响应

```
secret/astra-registry-cred created
```

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 `[your_registry_path]`。 `kube-rbac-proxy` 将映像推送到注册表路径中 [上一步](#)。
- c. 更改 `[your_registry_path]`。 `acc-operator-controller-manager` 将映像推送到注册表路径中 [上一步](#)。

```
<strong>astra_control_center_operator_deploy.yaml</strong>
```

```
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: [your_registry_path]/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: [your_registry_path]/acc-operator:23.04.36
          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
```

响应示例:

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.astra.
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	23.04.2-7

<code>astraAddress</code>

正在设置 ...	指导	Type	示例
<code>astraAddress</code>	<p>更改 <code>astraAddress</code> 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均衡器配置的FQDN或IP地址相同 "Astra 控制中心要求"。</p> <p>注意：请勿使用 <code>http://</code> 或 <code>https://</code> 地址中。复制此 FQDN 以在中使用 后续步骤。</p>	string	<code>astra.example.com</code>

<code>autoSupport</code>

您在本节中的选择将决定您是否要参与NetApp主动支持应用程序NetApp Active IQ 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
<code>autoSupport.enrolled</code>	两者之一 <code>enrolled</code> 或 <code>url</code> 必须选择字段	更改 <code>enrolled</code> 用于将AutoSupport连接到 <code>false</code> 对于不具有Internet连接或保留的站点 <code>true</code> 对于已连接站点。的设置 <code>true</code> 允许将匿名数据发送到NetApp以获得支持。默认选择为 <code>false</code> 和表示不会向NetApp发送任何支持数据。	布尔值	<code>false</code> (此值为默认值)
<code>autoSupport.url</code>	两者之一 <code>enrolled</code> 或 <code>url</code> 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
<code>imageRegistry.name</code>	Required	在中推送映像的映像注册表的名称 上一步 。请勿使用 <code>http://</code> 或 <code>https://</code> 注册表名称。	string	<code>example.registry.com/astra</code>
<code>imageRegistry.secret</code>	如果您为输入字符串、则为必填项 <code>imageRegistry.name</code> requires a secret. IMPORTANT: If you are using a registry that does not require authorization, you must delete this <code>secret</code> 行内 <code>imageRegistry</code> 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	<code>astra-registry-cred</code>

<code>storageClass</code>

正在设置 ...	指导	Type	示例
storageClass	<p>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 <code>kubectl get sc</code> 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-<code><version>.manifest</code>)、并将用于Astra PV。如果未设置、则会使用默认存储类。</p> <p>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。</p>	string	ontap-gold

<code>volumeReclaimPolicy</code>

正在设置 ...	指导	Type	选项
volumeReclaimPolicy	<p>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。</p>	string	<ul style="list-style-type: none">• Retain (这是默认值)• Delete

<code>ingressType</code>

正在设置 ...	指导	Type	选项
ingressType	<p>请使用以下入口类型之一：</p> <p>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。</p> <p>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。</p> <p>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均衡器、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均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 Astra 控制中心选择的 DNS 名称指向负载均衡的 IP 地址。</p> <p>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"。</p>	string	<ul style="list-style-type: none">• Generic (这是默认值)• AccTraefik

<code>scaleSize</code>

正在设置 ...	指导	Type	选项
scaleSize	<p>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 的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。</p> <p>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。</p>	string	<ul style="list-style-type: none">• Small• Medium (这是默认值)

<code>astraResourcesScaler</code>

正在设置 ...	指导	Type	选项
astraResourcesScaler	<p>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。</p> <p>但是、在使用较小的开发或测试集群的情况下、CR字段为 astraResourcesScaler 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。</p>	string	<ul style="list-style-type: none">• Default (这是默认值)• Off

<code>additionalValues</code>

- 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
<code>crds.externalCertManager</code>	<p>如果使用外部证书管理器、请进行更改 <code>externalCertManager to true</code>。默认值 <code>false</code> 使Astra控制中心在安装期间安装自己的证书管理器CRD。</p> <p>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。</p>	布尔值	False (此值为默认值)
<code>crds.externalTraefik</code>	<p>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。</p>	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。

```
<strong>astra_control_center.yaml</strong>
```

```
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: {}
  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]
```

响应示例:

```
namespace/netapp-acc created
```

2. 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]
```

响应示例:

```
astracontrolcenter.astra.netapp.io/astra created
```



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作用力控制中心。

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。

响应示例

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-78789d7549-1k686	1/1	Running	0
9h			
bucket-service-65c7d95496-24x7l	1/1	Running	3
(9h ago) 9h			
cert-manager-c9f9fbf9f-k8zq2	1/1	Running	0
9h			
cert-manager-c9f9fbf9f-qj1zm	1/1	Running	0
9h			
cert-manager-cainjector-dbbbd8447-b5q1l	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-mk917	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)为确保安装完成、您可以观看 `acc-operator` 使用以下命令记录。

```
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	23.04.2-7	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均衡

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以下传入控制器类型的示例步骤:

- Istio入口
- nginx 入口控制器
- OpenShift 入口控制器

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。

版权信息

版权所有 © 2023 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 <http://www.netapp.com/TM> 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。