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

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"。

您需要的内容

- "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。

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评估下载页面" 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

NetApp Astra kubectl命令行插件可节省执行与部署和升级Astra控制中心相关的常见任务所需的时间。

您需要的内容

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=22.11.0-82
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=22.11.0-82
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/22.11.0-82/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/22.11.0-82`）。

   ```
   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:[version x.y.z]
        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	`22.11.0-82`

<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
`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`

<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` 安装所需的其他Trident storageClass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Trident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"`)如果您不想配置入口控制器、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"loadbalancer"服务类型和入口的详细信息、请参见 "要求"。	string	• `Generic` (这是默认值) • `AccTraefik`

<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>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)

```
<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"
  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
```

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-76d8d845c9-ggds2                1/1     Running   0
14m
activity-6cc67ff9f4-z48mr                     1/1     Running   2
(8m32s ago)   9m
api-token-authentication-7s67v                1/1     Running   0
8m56s
api-token-authentication-bplb4                1/1     Running   0
8m56s
api-token-authentication-p2c9z                1/1     Running   0
8m56s
asup-6cdfbc6795-md8vn                         1/1     Running   0
9m14s
authentication-9477567db-8hnc9                1/1     Running   0
7m4s
bucketservice-f4dbdfcd6-wqzkw                 1/1     Running   0
8m48s
cert-manager-bb756c7c4-wm2cv                  1/1     Running   0
14m
cert-manager-cainjector-c9bb86786-8wrf5       1/1     Running   0
14m
cert-manager-webhook-dd465db99-j2w4x          1/1     Running   0
14m
certificates-68dff9cdd6-kcvml                 1/1     Running   2
(8m43s ago)   9m2s
certificates-68dff9cdd6-rsnsb                 1/1     Running   0
9m2s
cloud-extension-69d48c956c-2s8dt              1/1     Running   3
(8m43s ago)   9m24s
cloud-insights-service-7c4f48b978-7gvlh       1/1     Running   3
(8m50s ago)   9m28s
composite-compute-7d9ff5f68-nxbhl             1/1     Running   0
8m51s
composite-volume-57b4756d64-nl66d             1/1     Running   0
9m13s
credentials-6dbc55f89f-qpzff                  1/1     Running   0
11m
entitlement-67bfb6d7-gl6kp                    1/1     Running   4
(8m33s ago)   9m38s
features-856cc4dccc-mxbdb                     1/1     Running   0
9m20s
fluent-bit-ds-4rtsp                           1/1     Running   0
```

```
6m54s
fluent-bit-ds-9rqll                          1/1     Running   0
6m54s
fluent-bit-ds-w5mp7                          1/1     Running   0
6m54s
graphql-server-7c7cc49776-jz2kn              1/1     Running   0
2m29s
identity-87c59c975-9jpnf                     1/1     Running   0
9m6s
influxdb2-0                                  1/1     Running   0
13m
keycloak-operator-84ff6d59d4-qcnmc           1/1     Running   0
7m1s
krakend-cbf6c7df9-mdtzv                      1/1     Running   0
2m30s
license-5b888b78bf-plj6j                     1/1     Running   0
9m32s
login-ui-846b4664dd-fz8hv                    1/1     Running   0
2m24s
loki-0                                       1/1     Running   0
13m
metrics-facade-779cc9774-n26rw               1/1     Running   0
9m18s
monitoring-operator-974db78f-pkspq           2/2     Running   0
6m58s
nats-0                                       1/1     Running   0
13m
nats-1                                       1/1     Running   0
13m
nats-2                                       1/1     Running   0
13m
nautilus-7bdc7ddc54-49tfn                    1/1     Running   0
7m50s
nautilus-7bdc7ddc54-cwc79                    1/1     Running   0
9m36s
openapi-5584ff9f46-gbrdj                     1/1     Running   0
9m17s
openapi-5584ff9f46-z9mzk                     1/1     Running   0
9m17s
packages-bfc58cc98-lpxq9                     1/1     Running   0
8m58s
polaris-consul-consul-server-0               1/1     Running   0
13m
polaris-consul-consul-server-1               1/1     Running   0
13m
polaris-consul-consul-server-2               1/1     Running   0
```

```
13m
polaris-keycloak-0                          1/1     Running   3
(6m15s ago)   6m56s
polaris-keycloak-1                          1/1     Running   0
4m22s
polaris-keycloak-2                          1/1     Running   0
3m41s
polaris-keycloak-db-0                       1/1     Running   0
6m56s
polaris-keycloak-db-1                       1/1     Running   0
4m23s
polaris-keycloak-db-2                       1/1     Running   0
3m36s
polaris-mongodb-0                           2/2     Running   0
13m
polaris-mongodb-1                           2/2     Running   0
13m
polaris-mongodb-2                           2/2     Running   0
12m
polaris-ui-5ccff47897-8rzgh                 1/1     Running   0
2m33s
polaris-vault-0                             1/1     Running   0
13m
polaris-vault-1                             1/1     Running   0
13m
polaris-vault-2                             1/1     Running   0
13m
public-metrics-6cb7bfc49b-p54xm             1/1     Running   1
(8m29s ago)   9m31s
storage-backend-metrics-5c77994586-kjn48    1/1     Running   0
8m52s
storage-provider-769fdc858c-62w54           1/1     Running   0
8m54s
task-service-9ffc484c5-kx9f4                1/1     Running   3
(8m44s ago)   9m34s
telegraf-ds-bphb9                           1/1     Running   0
6m54s
telegraf-ds-rtsm2                           1/1     Running   0
6m54s
telegraf-ds-s9h5h                           1/1     Running   0
6m54s
telegraf-rs-lbpv7                           1/1     Running   0
6m54s
telemetry-service-57cfb998db-zjx78          1/1     Running   1
(8m40s ago)   9m26s
tenancy-5d5dfbcf9f-vmbxh                    1/1     Running   0
```

```
9m5s
traefik-7b87c4c474-jmgp2                    1/1     Running   0
2m24s
traefik-7b87c4c474-t9k8x                    1/1     Running   0
2m24s
trident-svc-c78f5b6bd-nwdsq                 1/1     Running   0
9m22s
vault-controller-55bbc96668-c6425           1/1     Running   0
11m
vault-controller-55bbc96668-lq9n9           1/1     Running   0
11m
vault-controller-55bbc96668-rfkgg           1/1     Running   0
11m
```

2. (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 。

3. 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  22.11.0-82  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

您需要的内容

- 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。

步骤

1. 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
```

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"。

=
:allow-uri-read: