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 Control Center，请从 NetApp 支持站点下载安装包，并执行以下步骤在您的环境中安装 Astra Control Center Operator 和 Astra Control Center。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。

对于Red Hat OpenShift环境、您可以使用 "备用操作步骤" 使用 OpenShift OperatorHub 安装 Astra Control Center 。

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"。
- 确保所有集群操作员均处于运行状况良好且可用。

```
kubectl get clusteroperators
```

- 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控制中心就不会安装自己的证书管理器。

关于此任务

Astra 控制中心安装过程将执行以下操作：

- 将Astra组件安装到中 `netapp-acc` (或自定义命名的)命名空间。
- 创建默认帐户。
- 建立默认管理用户电子邮件地址和默认一次性密码。系统会为该用户分配首次登录到UI所需的所有者角色。
- 帮助您确定所有 Astra 控制中心 Pod 是否正在运行。
- 安装 Astra UI 。

> (仅限适用场景 Astra数据存储早期访问计划("EAP")版本)如果要使用控制中心管理Astra数据存储并启用VMware工作流、请仅在上部署Astra控制中心 `pcloud` 命名空间、而不是 `netapp-acc` 此操作步骤 的步骤中介绍的命名空间或自定义命名空间。

> 请勿在整个安装过程中执行以下命令、以避免删除所有Astra控制中心Pod： `kubectl delete -f astra_control_center_operator_deploy.yaml`

> 如果您使用的是 Red Hat 的 Podman 而不是 Docker 引擎，则可以使用 Podman 命令代替 Docker 命令。

步骤

要安装 Astra 控制中心，请执行以下步骤：

- 下载并解包Astra Control Center软件包
- 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

下载并解包Astra Control Center软件包

1. 下载 Astra Control Center 捆绑包 (`astra-control-center-[version].tar.gz`) "NetApp 支持站点"。
2. 从下载 Astra 控制中心证书和密钥的 zip "NetApp 支持站点"。
3. （可选）使用以下命令验证捆绑包的签名：

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

4. 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为不同CPU架构和操作系统的插件提供二进制文件。在执行此任务之前、您需要了解您的CPU和操作系统。在Linux和Mac操作系统上、您可以使用 `uname -a` 用于收集此信息的命令。

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架构所需文件的名称：

```
ls kubectl-astra/
```

2. 将文件复制到与标准文件相同的位置 `kubectl` 实用程序。在此示例中、将显示 `kubectl` 实用程序位于中 `/usr/local/bin` 目录。替换 `<binary-name>` 使用所需文件的名称：

```
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Astra目录：

```
cd acc
```

2. [substep_image_local_registry_push]]将Astra控制中心映像目录中的软件包映像推送到本地注册表。在运行命令之前、请执行以下替换操作：
   - 将bundle_file替换为Astra Control捆绑包文件的名称(例如、 `acc.manifest.yaml`）。
   - 将my_regRegistry替换为Docker存储库的URL。
   - 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_REGISTRY
-u MY_REGISTRY_USER -p MY_REGISTRY_TOKEN
```

Podman

1. 登录到注册表：

```
podman login [your_registry_path]
```

2. 运行以下脚本、按照注释中的说明替换<your_registry>：

```
# You need to be at the root of the tarball.
# You should see these files to confirm correct location:
#   acc.manifest.yaml
#   acc/

# Replace <YOUR_REGISTRY> with your own registry (e.g
registry.customer.com or registry.customer.com/testing, etc..)
export REGISTRY=<YOUR_REGISTRY>
export PACKAGENAME=acc
export PACKAGEVERSION=22.08.1-26
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
  # Load to local cache
  astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image(s): //')

  # Remove path and keep imageName.
  astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')

  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/netapp/astra/${PACKAGENAME}
/${PACKAGEVERSION}/${astraImageNoPath}

  # Push to the local repo.
  podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/
${PACKAGEVERSION}/${astraImageNoPath}
done
```

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]
```

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/22.08.1-26`）。

```
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、则需要在重新创建命名空间后重新生成命名空间的密钥。

c. 创建 `netapp-acc` (或自定义命名)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)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
```

a. （可选）如果您希望集群在安装后由 Astra 控制中心自动管理，请确保在您要使用此命令部署到的 Astra 控制中心命名空间中提供 kubeconfig 作为机密：

```
kubectl create secret generic [acc-kubeconfig-cred or custom secret
name] --from-file=<path-to-your-kubeconfig> -n [netapp-acc or custom
namespace]
```

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。

d. （对于使用 Astra 数据存储预览版的安装）请参见有关的此已知问题描述 "存储类配置程序以及需要对 YAML 进行的其他更改"。

```
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
  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
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: [your_registry_path]/acc-operator:[version x.y.z]
        imagePullPolicy: IfNotPresent
      imagePullSecrets: []
```

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_min.yaml`)进行帐户、AutoSupport 、注册表和其他必要配置：

> `astra_control_center_min.yaml` 是默认CR、适用于大多数安装。熟悉所有内容 "CR选项及其潜在值" 以确保为您的环境正确部署Astra控制中心。如果您的环境需要其他自定义设置、您可以使用 `astra_control_center.yaml` 作为替代CR。

```
vim astra_control_center_min.yaml
```

> 如果您使用的注册表不需要授权、则必须删除 `secret` 行内 `imageRegistry` 否则安装将失败。

a. 更改 `[your_registry_path]` 到上一步中将映像推送到的注册表路径。

b. 更改 `accountName` 字符串、表示要与帐户关联的名称。

c. 更改 `astraAddress` 指向要在浏览器中用于访问Astra的FQDN的字符串。请勿使用 `http://` 或 `https://` 地址中。复制此 FQDN 以在中使用 后续步骤。

d. 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。

e. 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。

f. 如果使用外部证书管理器、请将以下行添加到 `spec`：

```
spec:
  crds:
    externalCertManager: true
```

g. (可选)添加名字 `firstName` 和姓氏 `lastName` 与帐户关联的用户的。您可以在用户界面中立即或稍后执行此步骤。

h. (可选)更改 `storageClass` 如果您的安装需要、则为另一个Trident storageClass资源指定值。

i. （可选）如果您希望集群在安装后由 Astra 控制中心自动管理，并且您已经这样了 已为此集群创建包含 kubeconfig 的密钥下、通过在此YAML文件中添加一个名为的新字段来提供密钥名称 `astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"`

j. 完成以下步骤之一：

- * 其他传入控制器（ ingressType： Generic ） *：这是 Astra 控制中心的默认操作。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。

  默认的Astra Control Center安装用于设置其网关 (`service/traefik`)的类型 `ClusterIP`。此默认安装要求您另外设置一个 Kubernetes IngressController/Ingress ，以便向其路由流量。如果要使用入口，请参见 "设置传入以进行负载平衡"。

- 服务负载平衡器**(ingressType：AccTraefik)**：如果您不想安装IngressController或创建Ingress资源、请设置 `ingressType` to `AccTraefik`。

  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。

  ⓘ 有关 "loadbalancer" 服务类型和入口的详细信息，请参见 "要求"。

```
apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "Example"
  astraVersion: "ASTRA_VERSION"
  astraAddress: "astra.example.com"
  astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"
  ingressType: "Generic"
  autoSupport:
    enrolled: true
  email: "[admin@example.com]"
  firstName: "SRE"
  lastName: "Admin"
  imageRegistry:
    name: "[your_registry_path]"
    secret: "astra-registry-cred"
  storageClass: "ontap-gold"
```

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_min.yaml -n [netapp-acc or custom
namespace]
```

响应示例：

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

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。

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-6b44d68d94-d8m55              1/1     Running   0
13m
activity-78f99ddf8-hltct                    1/1     Running   0
10m
api-token-authentication-457nl              1/1     Running   0
9m28s
api-token-authentication-dgwsz              1/1     Running   0
9m28s
api-token-authentication-hmqqc              1/1     Running   0
9m28s
asup-75fd554dc6-m6qzh                       1/1     Running   0
9m38s
authentication-6779b4c85d-92gds             1/1     Running   0
8m11s
bucketservice-7cc767f8f8-lqwr8              1/1     Running   0
9m31s
certificates-549fd5d6cb-5kmd6               1/1     Running   0
9m56s
certificates-549fd5d6cb-bkjh9               1/1     Running   0
9m56s
cloud-extension-7bcb7948b-hn8h2             1/1     Running   0
10m
cloud-insights-service-56ccf86647-fgg69     1/1     Running   0
9m46s
composite-compute-677685b9bb-7vgsf          1/1     Running   0
10m
composite-volume-657d6c5585-dnq79           1/1     Running   0
9m49s
credentials-755fd867c8-vrlmt                1/1     Running   0
11m
entitlement-86495cdf5b-nwhh2                1/1     Running   2
10m
features-5684fb8b56-8d6s8                   1/1     Running   0
10m
fluent-bit-ds-rhx7v                         1/1     Running   0
7m48s
fluent-bit-ds-rjms4                         1/1     Running   0
7m48s
fluent-bit-ds-zf5ph                         1/1     Running   0
7m48s
graphql-server-66d895f544-w6hjd             1/1     Running   0
```

```
3m29s
identity-744df448d5-rlcmm                1/1     Running   0
10m
influxdb2-0                              1/1     Running   0
13m
keycloak-operator-75c965cc54-z7csw       1/1     Running   0
8m16s
krakend-798d6df96f-9z2sk                 1/1     Running   0
3m26s
license-5fb7d75765-f8mjg                 1/1     Running   0
9m50s
login-ui-7d5b7df85d-l2s7s                1/1     Running   0
3m20s
loki-0                                   1/1     Running   0
13m
metrics-facade-599b9d7fcc-gtmgl          1/1     Running   0
9m40s
monitoring-operator-67cc74f844-cdplp     2/2     Running   0
8m11s
nats-0                                   1/1     Running   0
13m
nats-1                                   1/1     Running   0
13m
nats-2                                   1/1     Running   0
12m
nautilus-769f5b74cd-k5jxm                1/1     Running   0
9m42s
nautilus-769f5b74cd-kd9gd                1/1     Running   0
8m59s
openapi-84f6ccd8ff-76kvp                 1/1     Running   0
9m34s
packages-6f59fc67dc-4g2f5                1/1     Running   0
9m52s
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   0
8m7s
polaris-keycloak-1                       1/1     Running   0
5m49s
polaris-keycloak-2                       1/1     Running   0
5m15s
polaris-keycloak-db-0                    1/1     Running   0
```

```
8m6s
polaris-keycloak-db-1                     1/1     Running   0
5m49s
polaris-keycloak-db-2                     1/1     Running   0
4m57s
polaris-mongodb-0                         2/2     Running   0
13m
polaris-mongodb-1                         2/2     Running   0
12m
polaris-mongodb-2                         2/2     Running   0
12m
polaris-ui-565f56bf7b-zwr8b               1/1     Running   0
3m19s
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-6d86d66444-2wbzl           1/1     Running   0
9m30s
storage-backend-metrics-77c5d98dcd-dbhg5 1/1     Running   0
9m44s
storage-provider-78c885f57c-6zcv4         1/1     Running   0
9m36s
telegraf-ds-2l2m9                         1/1     Running   0
7m48s
telegraf-ds-qfzgh                         1/1     Running   0
7m48s
telegraf-ds-shrms                         1/1     Running   0
7m48s
telegraf-rs-bjpkt                         1/1     Running   0
7m48s
telemetry-service-6684696c64-qzfdf        1/1     Running   0
10m
tenancy-6596b6c54d-vmpsm                  1/1     Running   0
10m
traefik-7489dc59f9-6mnst                  1/1     Running   0
3m19s
traefik-7489dc59f9-xrkgg                  1/1     Running   0
3m4s
trident-svc-6c8dc458f5-jswcl              1/1     Running   0
10m
vault-controller-6b954f9b76-gz9nm         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。如果日志中指示集群注册失败，您可以通过添加集群工作流再次尝试注册 "在 UI 中" 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一次性密码：

```
kubectl get AstraControlCenter -n netapp-acc
```

响应：

```
NAME     UUID                                       VERSION      ADDRESS
READY
astra    ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f   22.08.1-26
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 Kubernetes 入口控制器，用于管理对服务的外部访问，例如集群中的负载平衡。

此操作步骤 介绍了如何设置入口控制器 (`ingressType:Generic`）。这是 Astra 控制中心的默认操作。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。

> 如果您不想设置入口控制器、可以设置 `ingressType:AccTraefik`)。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 "loadbalancer" 服务类型和入口的详细信息，请参见 "要求"。

根据您使用的入口控制器类型，步骤会有所不同：

- 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。
- 您使用的是介于 v1.19 和 v1.22 之间的 Kubernetes 版本，包括 v1.19 和 v1.22 。

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. 在中部署传入资源 `netapp-acc` (或自定义命名)命名空间使用v1beta1 (在Kubernetes版本低于或1.22的情况下已弃用)或v1资源类型作为已弃用或新模式：

   输出：

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: IngressClass
metadata:
  name: istio
spec:
  controller: istio.io/ingress-controller
---
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress
  namespace: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          serviceName: traefik
          servicePort: 80
```

对于v1新模式、请遵循以下示例：

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

输出：

```
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: istio-system
spec:
  ingressClassName: istio
  tls:
  - hosts:
    - <ACC addess>
    secretName: [tls secret name]
  rules:
  - host: [ACC addess]
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: traefik
            port:
              number: 80
```

5. 照常部署Astra控制中心。
6. 检查入口状态：

```
kubectl get ingress -n netapp-acc
```

响应：

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

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 `v1beta1` (在低于或1.22的Kubernetes版本中已弃用)或 `v1` 已弃用或新模式的资源类型：

   a. 对于A `v1beta1` 已弃用模式、请按照以下示例进行操作：

```
apiVersion: extensions/v1beta1
Kind: IngressClass
metadata:
  name: ingress-acc
  namespace: [netapp-acc or custom namespace]
  annotations:
    kubernetes.io/ingress.class: [class name for nginx controller]
spec:
  tls:
  - hosts:
    - <ACC address>
    secretName: [tls secret name]
  rules:
  - host: [ACC address]
    http:
      paths:
      - backend:
        serviceName: traefik
        servicePort: 80
        pathType: ImplementationSpecific
```

   b. 。 `v1` 新架构、请按照以下示例进行操作：

```yaml
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 addess>
    http:
      paths:
        - path:
          backend:
            service:
              name: traefik
              port:
                number: 80
          pathType: ImplementationSpecific
```

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、输入在中使用的FQDN `astraAddress` 在中 `astra_control_center_min.yaml` CR时间您安装了 Astra 控制中心。
2. 出现提示时接受自签名证书。

> 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_min.yaml` CR时间 您安装了 Astra 控制中心、后跟一次性密码 (`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。

> 如果您是首次登录，但忘记了密码，并且尚未创建任何其他管理用户帐户，请联系 NetApp 支持部门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 --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name) -c manager
```

下一步行动

执行以完成部署 "设置任务"。

=
:allow-uri-read:

了解POD安全策略限制

Astra控制中心通过POD安全策略(PSP)支持权限限制。通过POD安全策略、您可以限制哪些用户或组能够运行容器以及这些容器可以具有哪些权限。

某些Kubernetes分发版(例如RKE2)的默认POD安全策略限制性过强、在安装Astra Control Center时会出现问题。

您可以使用此处提供的信息和示例来了解Astra控制中心创建的POD安全策略、并配置POD安全策略、以便在不干扰Astra控制中心功能的情况下提供所需的保护。

由Astra控制中心安装的Psps

Astra控制中心会在安装期间创建多个POD安全策略。其中一些是永久性的、其中一些是在某些操作期间创建的、操作完成后会将其删除。

在安装期间创建的**Psps**

在安装Astra控制中心期间、Astra控制中心操作员会安装一个自定义POD安全策略、一个角色对象和一个RoleBinding.对象、以支持在Astra控制中心命名空间中部署Astra控制中心服务。

新策略和对象具有以下属性：

```
kubectl get psp

NAME                            PRIV    CAPS          SELINUX    RUNASUSER
FSGROUP     SUPGROUP    READONLYROOTFS   VOLUMES
avp-psp                         false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *
netapp-astra-deployment-psp     false                 RunAsAny   RunAsAny
RunAsAny    RunAsAny    false            *

kubectl get role

NAME                                     CREATED AT
netapp-astra-deployment-role             2022-06-27T19:34:58Z

kubectl get rolebinding

NAME                                     ROLE
AGE
netapp-astra-deployment-rb               Role/netapp-astra-deployment-role
32m
```

备份操作期间创建的**Psps**

在备份操作期间、Astra控制中心会创建一个动态POD安全策略、一个ClusterRole对象和一个RoleBinding.它们支持备份过程、该过程会在单独的命名空间中进行。

新策略和对象具有以下属性：

```
kubectl get psp

NAME                            PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-astra-backup             false   DAC_READ_SEARCH
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false            *

kubectl get role

NAME                  CREATED AT
netapp-astra-backup   2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                  ROLE                        AGE
netapp-astra-backup   Role/netapp-astra-backup   62s
```

在集群管理期间创建的**Psps**

管理集群时、Astra控制中心会在受管集群中安装NetApp监控操作员。此运算符将创建Pod安全策略、ClusterRole对象和RoleBinding对象、以便在Astra Control Center命名空间中部署遥测服务。

新策略和对象具有以下属性：

```
kubectl get psp

NAME                            PRIV    CAPS
SELINUX    RUNASUSER          FSGROUP     SUPGROUP    READONLYROOTFS
VOLUMES
netapp-monitoring-psp-nkmo      true    AUDIT_WRITE,NET_ADMIN,NET_RAW
RunAsAny   RunAsAny           RunAsAny    RunAsAny    false            *

kubectl get role

NAME                                          CREATED AT
netapp-monitoring-role-privileged             2022-07-21T00:00:00Z

kubectl get rolebinding

NAME                                                  ROLE
AGE
netapp-monitoring-role-binding-privileged             Role/netapp-
monitoring-role-privileged              2m5s
```

在命名空间之间启用网络通信

某些环境使用NetworkPolicy构造来限制命名空间之间的流量。Astra控制中心操作员、Astra控制中心和适用于VMware vSphere的Astra插件都位于不同的命名空间中。这些不同命名空间中的服务需要能够彼此通信。要启用此通信、请执行以下步骤。

步骤

1. 删除Astra控制中心命名空间中的任何NetworkPolicy资源：

```
kubectl get networkpolicy -n netapp-acc
```

2. 对于上述命令返回的每个NetworkPolicy对象、请使用以下命令将其删除。将<object_name>替换为返回对象的名称：

```
kubectl delete networkpolicy <OBJECT_NAME> -n netapp-acc
```

3. 应用以下资源文件以配置Acc-AVP-network-policy对象、以允许适用于VMware vSphere的Astra插件服务向Astra控制中心服务发出请求。将括号<>中的信息替换为您环境中的信息：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-avp-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <PLUGIN_NAMESPACE_NAME> #
REPLACE THIS WITH THE ASTRA PLUGIN FOR VMWARE VSPHERE NAMESPACE NAME
```

4. 应用以下资源文件来配置Acc-operator-network-policy对象、以使Astra控制中心操作员能够与Astra控制中心服务进行通信。将括号<>中的信息替换为您环境中的信息：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: acc-operator-network-policy
  namespace: <ACC_NAMESPACE_NAME> # REPLACE THIS WITH THE ASTRA CONTROL
CENTER NAMESPACE NAME
spec:
  podSelector: {}
  policyTypes:
    - Ingress
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: <NETAPP-ACC-OPERATOR> #
REPLACE THIS WITH THE OPERATOR NAMESPACE NAME
```

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。您需要将其从计划安装Astra控制中心的命名空间中删除。

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NetApp-Accc命名空间中获取资源配额：

```
kubectl get quota -n netapp-acc
```

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
```

```
kubectl delete resourcequota  pods-low -n netapp-acc
```

```
kubectl delete resourcequota  pods-medium -n netapp-acc
```

3. 在NetApp-Accc命名空间中获取限制范围：

```
kubectl get limits -n netapp-acc
```

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
```

=
:allow-uri-read: