Astra Control Center をインストールします

Astra Control Center

NetApp
November 21, 2023

目次

標準の手順で Astra Control Center をインストールします

Astra Control Center をインストールするには、NetApp Support Siteからインストールバンドルをダウンロードし、次の手順を実行して、 Astra Control Center Operator と Astra Control Center を環境にインストールします。この手順を使用して、インターネット接続環境またはエアギャップ環境に Astra コントロールセンターをインストールできます。

Red Hat OpenShift環境では、を使用できます "代替手順" OpenShift OperatorHub を使用して Astra Control Center をインストールします。

必要なもの

- "インストールを開始する前に、 Astra Control Center の導入環境を準備します"。
- 使用環境でポッドセキュリティポリシーを設定または設定したい場合は、ポッドセキュリティポリシーと、それらがAstra Control Centerのインストールに与える影響について理解しておいてください。を参照してください "ポッドのセキュリティポリシーの制限事項を理解します"。
- すべてのクラスタオペレータが正常な状態であり、使用可能であることを確認します。

```
kubectl get clusteroperators
```

- すべての API サービスが正常な状態であり、使用可能であることを確認します。

```
kubectl get apiservices
```

- 使用するネットアップFQDNがこのクラスタにルーティング可能であることを確認します。つまり、内部 DNS サーバに DNS エントリがあるか、すでに登録されているコア URL ルートを使用しています。
- クラスタにcert-managerがすでに存在する場合は、いくつかを実行する必要があります "事前に必要な手順" そのため、Astra Control Centerは独自の証明書管理ツールをインストールしません。

このタスクについて

Astra Control Center のインストールプロセスでは、次のことが実行されます。

- Astraコンポーネントをにインストールします `netapp-acc` （またはカスタム名）ネームスペース。
- デフォルトアカウントを作成します。
- デフォルトの管理ユーザのEメールアドレスとデフォルトのワンタイムパスワードを設定します。このユーザには、UIへの初回ログインに必要なシステムのオーナーロールが割り当てられます。
- Astra Control Center のすべてのポッドが実行されていることを確認するのに役立ちます。
- Astra の UI をインストールします。

> （環境 the Astra Data Store Early Access Program（EAP）リリースのみ）Astra Control Center を使用してAstraデータストアを管理し、VMwareワークフローを有効にする場合は、Astra Control Centerのみをに導入します `pcloud` ではなく、ネームスペースで指定します `netapp-acc` この手順 の手順で説明するネームスペースまたはカスタムネームスペース。

すべてのAstra Control Centerポッドが削除されないようにするため、インストールプロセス全体で次のコマンドを実行しないでください。 `kubectl delete -f astra_control_center_operator_deploy.yaml`

Docker Engine の代わりに Red Hat の Podman を使用している場合は、 Docker コマンドの代わりに Podman コマンドを使用できます。

手順

Astra Control Center をインストールするには、次の手順に従います。

- Astra Control Centerバンドルをダウンロードして開梱します
- ネットアップAstra kubectlプラグインをインストール
- [イメージをローカルレジストリに追加します]
- [認証要件を持つレジストリのネームスペースとシークレットを設定します]
- Astra Control Center オペレータを設置します
- Astra Control Center を設定します
- Astra Control Center とオペレータのインストールを完了します
- [システムステータスを確認します]
- [ロードバランシング用の入力を設定します]
- Astra Control Center UI にログインします

Astra Control Centerバンドルをダウンロードして開梱します

1. Astra Control Center バンドルをダウンロードします (`astra-control-center-[version].tar.gz`) をクリックします "NetApp Support Site"。
2. から Astra Control Center 証明書とキーの zip をダウンロードします "NetApp Support Site"。
3. （任意）次のコマンドを使用して、バンドルのシグニチャを確認します。

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

4. 画像を抽出します。

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

ネットアップAstra kubectlプラグインをインストール

ネットアップアストラ `kubectl` コマンドラインプラグインにより、Astra Control Centerの導入とアップグレ

ードに関連する一般的なタスクを実行する時間を節約できます。

必要なもの

ネットアップでは、プラグイン用のバイナリを提供しており、CPUアーキテクチャやオペレーティングシステムが異なる場合はそのプラグインをこのタスクを実行する前に、使用しているCPUとオペレーティングシステムを把握しておく必要があります。LinuxおよびMacオペレーティングシステムでは、を使用できます `uname -a` この情報を収集するためのコマンドです。

手順

1. 使用可能なネットアップAstraを選択します `kubectl` プラグインバイナリ。オペレーティングシステムとCPUアーキテクチャに必要なファイルの名前をメモします。

```
ls kubectl-astra/
```

2. ファイルを標準と同じ場所にコピーします `kubectl` ユーティリティ。この例では、を使用しています `kubectl` ユーティリティはにあります `/usr/local/bin` ディレクトリ。交換してください `<binary-name>` 必要なファイル名を使用して、次の操作を実行します。

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

イメージをローカルレジストリに追加します

1. コンテナエンジンに応じた手順を実行します。

Docker です

1. Astraディレクトリに移動します。

```
cd acc
```

2. [[[[</Z1>[</Z1>[</Z1>[</Z1>[</Z1>[</Z1>[</Z1>[</Z1></Z1>_image_local_registry_push]]]]]]]]]</Z2>アストラControl Centerイメージディレクトリ内のパッケージイメージをローカルレジストリにプッシュします。</Z3>コマンドを実行する前に、次の置き換えを行ってください。
   - bundle_fileをAstra Controlバンドルファイルの名前に置き換えます（例： `acc.manifest.yaml` ）。
   - my_registryをDockerリポジトリのURLに置き換えます。
   - my_registry_userをユーザー名に置き換えます。
   - my_registry_tokenをレジストリの認証済みトークンに置き換えます。

```
kubectl astra packages push-images -m BUNDLE_FILE -r MY_REGISTRY
-u MY_REGISTRY_USER -p MY_REGISTRY_TOKEN
```

ポドマン

1. レジストリにログインします。

```
podman login [your_registry_path]
```

2. 次のスクリプトを実行して、コメントに記載されているように<your _registry>を置き換えます。

```
# 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
```

認証要件を持つレジストリのネームスペースとシークレットを設定します

1. Astra Control Centerホストクラスタ用のKUBECONFIGをエクスポートします。

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

2. 認証が必要なレジストリを使用する場合は、次の手順を実行する必要があります。

   a. を作成します `netapp-acc-operator` ネームスペース：

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

対応：

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

b. のシークレットを作成します `netapp-acc-operator` ネームスペース：Docker 情報を追加して次のコマンドを実行します。

> プレースホルダ `your_registry_path` 以前にアップロードした画像の場所と一致する必要があります（例： `[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]
```

回答例：

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

> シークレットの生成後にネームスペースを削除する場合は、ネームスペースが再作成されたあとにネームスペースのシークレットを再生成する必要があります。

c. を作成します `netapp-acc` （またはカスタムの名前付き）ネームスペース。

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

回答例：

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

d. のシークレットを作成します `netapp-acc` （またはカスタムの名前付き）ネームスペース。Docker 情報を追加して次のコマンドを実行します。

```
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]
```

応答

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

a. [[[[sup_kubeconfig _secret]] （オプション）インストール後に Astra Control Center でクラスタを自動的に管理する場合は、このコマンドを使用して展開する Astra Control Center ネームスペース内のシークレットとして kubeconfig を指定する必要があります。

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

Astra Control Center オペレータを設置します

1. ディレクトリを変更します。

```
cd manifests
```

2. Astra Control Centerオペレータ配置YAMLを編集します (`astra_control_center_operator_deploy.yaml`)を参照して、ローカルレジストリとシークレットを参照してください。

```
vim astra_control_center_operator_deploy.yaml
```

> 注釈付きサンプルYAMLは以下の手順に従います。

a. 認証が必要なレジストリを使用する場合は、のデフォルト行を置き換えます `imagePullSecrets: []` 次の条件を満たす場合：

```
imagePullSecrets:
- name: <astra-registry-cred>
```

b. 変更 `[your_registry_path]` をクリックします `kube-rbac-proxy` でイメージをプッシュしたレジストリパスへのイメージ 前の手順。

c. 変更 `[your_registry_path]` をクリックします `acc-operator-controller-manager` でイメージをプッシュしたレジストリパスへのイメージ 前の手順。

d. （Astra データストアプレビューを使用するインストールの場合）この問題に関する既知の情報を参照してください "ストレージクラスのプロビジョニングと YAML に対する追加の変更"。

```
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. Astra Control Center オペレータをインストールします。

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

回答例：

```
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. ポッドが実行中であることを確認します

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

Astra Control Center を設定します

1. Astra Control Centerカスタムリソース（CR）ファイルを編集します (`astra_control_center_min.yaml`)アカウント、AutoSupport 、レジストリ、およびその他の必要な設定を行うには、次の手順を実行します。

> `astra_control_center_min.yaml` はデフォルトのCRで、ほとんどのインストールに適しています。すべてのことをよく理解してください "CRオプションとその可能性のある値" お客様の環境にAstra Control Centerを正しく導入できるようにするため。環境で追加のカスタマイズが必要な場合は、を使用できます `astra_control_center.yaml` 代替CRとして。

```
vim astra_control_center_min.yaml
```

> 許可が不要なレジストリを使用している場合は、を削除する必要があります `secret` ラインの内側 `imageRegistry` または、インストールが失敗します。

a. 変更 `[your_registry_path]` 前の手順でイメージをプッシュしたレジストリパスに移動します。

b. を変更します `accountName` stringには、アカウントに関連付ける名前を指定します。

c. を変更します `astraAddress` ブラウザからAstraにアクセスする際に使用するFQDNを文字列で指定します。使用しないでください `http://` または `https://` をクリックします。この FQDN をコピーしてで使用します 後の手順。

d. を変更します `email` デフォルトの初期管理者アドレスを表す文字列。この E メールアドレスをコピーしてで使用します 後の手順。

e. 変更 `enrolled` を選択しますAutoSupport `false` インターネットに接続されていないか、または保持されているサイト `true` 接続されているサイト用。

f. 外部証明書マネージャを使用する場合は、に次の行を追加します `spec`：

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

g. （オプション）名を追加します `firstName` 姓を入力します `lastName` アカウントに関連付けられているユーザのこの手順は、 UI ですぐに実行することもあとで実行することもできます。

h. （オプション）を変更します `storageClass` インストールに必要な場合、別のTridentストレージクラスリソースへの値。

i. （オプション）インストール後に Astra Control Center でクラスタを自動的に管理する場合は このクラスタの kubeconfig を含むシークレットを作成しましたをクリックし、という名前のYAMLファイルに新しいフィールドを追加して、シークレットの名前を指定します `astraKubeConfigSecret: "acc-kubeconfig-cred or custom secret name"`

j. 次のいずれかの手順を実行します。

- * その他の入力コントローラ（ ingressType: Generic ） * ：これはアストラコントロールセンターでのデフォルトのアクションです。Astra Control Center を展開したら、 Astra Control Center を URL で公開するように入力コントローラを設定する必要があります。

  デフォルトのAstra Control Centerインストールでは、ゲートウェイがセットアップされます（`service/traefik`）を入力します `ClusterIP`。このデフォルトのインストールでは、トラフィックをルーティングするために Kubernetes IngressController/Ingress を追加で設定する必要があります。入力を使用する場合は、を参照してください "ロードバランシング用の入力を設定します"。

- サービスロードバランサ（**ingressType: AccTrafik**）: IngressControllerをインストールしない場合、または入力リソースを作成しない場合は、を設定します `ingressType` 終了：`AccTraefik`。

  これにより、Astra Control Centerが導入されます `traefik` Gateway as a Kubernetes LoadBalancer type serviceの略。

  Astra Control Centerは、タイプ「LoadBalancer」のサービスを使用します。（`svc/traefik` Astra Control Centerの名前空間）で、アクセス可能な外部IPアドレスが割り当てられている必要があります。お使いの環境でロードバランサが許可されていて、設定されていない場合は、MetalLB または別の外部サービスロードバランサを使用して、外部 IP アドレスをサービスに割り当てることができます。内部 DNS サーバ構成では、 Astra Control Center に選択した DNS 名を、負荷分散 IP アドレスに指定する必要があります。

サービスタイプ「 LoadBalancer 」および入力の詳細については、を参照してください "要件"。

```
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"
```

Astra Control Center とオペレータのインストールを完了します

1. 前の手順でまだ行っていない場合は、を作成します `netapp-acc` （またはカスタム）ネームスペース：

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

回答例：

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

2. にAstra Control Centerをインストールします `netapp-acc` （またはカスタムの）ネームスペース：

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

回答例：

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

システムステータスを確認します

OpenShift を使用する場合は、同等の OC コマンドを検証手順に使用できます。

1. すべてのシステムコンポーネントが正常にインストールされたことを確認します。

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

各ポッドのステータスがになっている必要があります `Running`。システムポッドが展開されるまでに数分かかることがあります。

回答例

```
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. （オプション）インストールが完了したことを確認するには、を参照してください `acc-operator` 次のコマンドを使用してログを作成します。

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

> `accHost` クラスタの登録は最後の処理の1つです。登録に失敗しても原因 の導入は失敗しません。ログにクラスタ登録エラーが示された場合は、クラスタ追加ワークフローを通じて再度登録を試行できます "UI で" または API 。

3. すべてのポッドが実行中の場合は、インストールが正常に完了したことを確認します (`READY` はです `True`）を使用してAstra Control Centerにログインする際に使用するワンタイムパスワードを取得します。

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

対応：

```
NAME    UUID                                        VERSION     ADDRESS
READY
astra   ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f  22.08.1-26
10.111.111.111  True
```

> UUIDの値をコピーします。パスワードはです `ACC-` 続けてUUIDの値を指定します (`ACC-[UUID]` または、この例では、 `ACC-9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f`）。

ロードバランシング用の入力を設定します

Kubernetes 入力コントローラをセットアップして、クラスタのロードバランシングなどのサービスへの外部アクセスを管理できます。

この手順 では、入力コントローラの設定方法について説明します (`ingressType:Generic`）。これは、Astra Control Center でのデフォルトのアクションです。Astra Control Center を展開したら、 Astra Control Center を URL で公開するように入力コントローラを設定する必要があります。

> 入力コントローラを設定しない場合は、を設定できます `ingressType:AccTraefik`)。Astra Control Centerは、タイプ「LoadBalancer」のサービスを使用します。（`svc/traefik` Astra Control Centerの名前空間）で、アクセス可能な外部IPアドレスが割り当てられている必要があります。お使いの環境でロードバランサが許可されていて、設定されていない場合は、MetalLB または別の外部サービスロードバランサを使用して、外部 IP アドレスをサービスに割り当てることができます。内部 DNS サーバ構成では、 Astra Control Center に選択した DNS 名を、負荷分散 IP アドレスに指定する必要があります。サービスタイプ「 LoadBalancer 」および入力の詳細については、を参照してください "要件"。

この手順は、使用する入力コントローラのタイプによって異なります。

- Istio入力
- nginx 入力コントローラ
- OpenShift 入力コントローラ

必要なもの

- が必要です "入力コントローラ" すでに導入されている必要があります。
- 。 "入力クラス" 入力コントローラに対応するものがすでに作成されている必要があります。
- V1.19 と v1.22 の間で Kubernetes のバージョンを使用している。

Istio Ingressの手順

1. Istio Ingressを設定します。

   この手順 では、「デフォルト」の構成プロファイルを使用してIstioが導入されていることを前提としています。

2. 入力ゲートウェイに必要な証明書と秘密鍵ファイルを収集または作成します。

   CA署名証明書または自己署名証明書を使用できます。共通名はAstraアドレス（FQDN）である必要があります。

   コマンド例：

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

3. シークレットを作成します `tls secret name` を入力します `kubernetes.io/tls` でTLS秘密鍵と証明書を使用する場合 `istio-system namespace` TLSシークレットで説明されているように、

   コマンド例：

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

   シークレットの名前はと一致する必要があります `spec.tls.secretName` で提供されます `istio-ingress.yaml` ファイル。

4. 入力リソースをに配置します `netapp-acc` （またはカスタムネームド）ネームスペースで、v1beta1（Kubernetesバージョンで1.22未満で廃止）またはv1リソースタイプを使用して、非推奨または新しいスキーマのいずれかに対応します。

   出力：

```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
```

v1の新しいスキーマについては、次の例を参照してください。

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

出力：

```
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. 通常どおりAstra Control Centerを導入します。
6. 入力のステータスを確認します。

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

対応：

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

Nginx Ingress Controller の手順

1. タイプのシークレットを作成します[`kubernetes.io/tls`]をクリックします `netapp-acc` （またはカス

タム名前付き）ネームスペース。を参照してください "TLS シークレット"。

2. 入力リソースをに配置します `netapp-acc` （またはカスタムの名前付き）ネームスペースのいずれかを使用します `v1beta1` （Kubernetesのバージョンが1.22より前の場合は廃止）または `v1` 非推奨または新しいスキーマのリソースタイプ：

   a. に設定します `v1beta1` 廃止されたスキーマの例を次に示します。

```
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. をクリックします `v1` 新しいスキーマの例を次に示します。

```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 入力コントローラの手順

1. 証明書を調達し、 OpenShift ルートで使用できるようにキー、証明書、および CA ファイルを取得します。
2. OpenShift ルートを作成します。

```
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
```

Astra Control Center UI にログインします

Astra Control Center をインストールした後、デフォルトの管理者のパスワードを変更し、 Astra Control Center UI ダッシュボードにログインします。

手順

1. ブラウザで、で使用したFQDNを入力します `astraAddress` を参照してください `astra_control_center_min.yaml` CR When（時間） Astra Control Center をインストールした。
2. プロンプトが表示されたら、自己署名証明書を受け入れます。

カスタム証明書はログイン後に作成できます。

3. Astra Control Centerのログインページで、に使用した値を入力します `email` インチ `astra_control_center_min.yaml` CR When（時間） Astra Control Center をインストールしたをクリックし、続けてワンタイムパスワードを入力します (`ACC-[UUID]`）。

   誤ったパスワードを 3 回入力すると、管理者アカウントは 15 分間ロックされます。

4. [**Login**] を選択します。
5. プロンプトが表示されたら、パスワードを変更します。

   初めてログインする際にパスワードを忘れた場合、他の管理ユーザアカウントがまだ作成されていないときは、ネットアップのサポートに問い合わせて、パスワードのリカバリに関するサポートを依頼してください。

6. （オプション）既存の自己署名 TLS 証明書を削除して、に置き換えます "認証局（ CA ）が署名したカスタム TLS 証明書"。

インストールのトラブルシューティングを行います

いずれかのサービスがにある場合 `Error` ステータスを確認すると、ログを調べることができます。400 ~ 500 の範囲の API 応答コードを検索します。これらは障害が発生した場所を示します。

手順

1. Astra Control Center のオペレータログを調べるには、次のように入力します。

```
kubectl logs --follow -n netapp-acc-operator $(kubectl get pods -n
netapp-acc-operator -o name) -c manager
```

次のステップ

を実行して導入を完了します "セットアップのタスク"。

=
:allow-uri-read:

ポッドのセキュリティポリシーの制限事項を理解します

Astra Control Centerは、PoDセキュリティポリシー（PSP）による特権制限をサポートします。ポッドセキュリティポリシーを使用すると、ユーザまたはグループがコンテナを実行できる対象や、それらのコンテナに付与できる権限を制限できます。

RKE2などの一部のKubernetesディストリビューションには、デフォルトのポッドセキュリティポリシーが用意されていますが、制限が厳しく、Astra Control Centerのインストール時に問題が発生します。

ここに記載されている情報と例を使用して、Astra Control Centerによって作成されるポッドセキュリティポリシーを理解し、Astra Control Centerの機能を妨げずに必要な保護を提供するポッドセキュリティポリシーを設定できます。

Astra Control Centerによってインストールされた**PSP**

Astra Control Centerは、インストール中にポッドのセキュリティポリシーをいくつか作成します。これらの中には永続的なものもあれば、一部のものは特定の処理中に作成され、処理が完了すると削除されます。

インストール中に作成された**PSP**

Astra Control Centerのインストール中に、Astra Control Centerオペレータは、Astra Control Centerネームスペースへの Astra Control Centerサービスの展開をサポートするために、カスタムのPoDセキュリティポリシー、Roleオブジェクト、およびRoleBindingオブジェクトをインストールします。

新しいポリシーとオブジェクトには次の属性があります。

```
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
```

バックアップ処理中に作成された**PSP**

バックアップ処理中に、Astra Control Centerは、動的なPODセキュリティポリシー、ClusterRoleオブジェクト、およびRoleBindingオブジェクトを作成します。これらの機能により、別のネームスペースで実行されるバックアッププロセスがサポートされます。

新しいポリシーとオブジェクトには次の属性があります。

```
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
```

クラスタ管理中に作成された**PSP**

クラスターを管理する場合、Astra Control Centerは管理対象クラスタにNetApp Monitoringオペレータをインストールします。この演算子は、PODセキュリティポリシー、ClusterRoleオブジェクト、およびRoleBindingオブジェクトを作成して、Astra Control Center名前空間にテレメトリサービスを展開します。

新しいポリシーとオブジェクトには次の属性があります。

```
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
```

ネームスペース間のネットワーク通信を有効にします

一部の環境では、NetworkPolicy構造体を使用してネームスペース間のトラフィックを制限します。Astra Control Centerオペレータ、Astra Control Center、およびAstra Plugin for VMware vSphereはすべて異なる名前空間に存在します。これらの異なるネームスペース内のサービスは、相互に通信できる必要があります。この通信をイネーブルにするには、次の手順を実行します。

手順

1. Astra Control Center名前空間に存在するNetworkPolicyリソースをすべて削除します。

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

2. 前のコマンドで返された各NetworkPolicyオブジェクトについて、次のコマンドを使用して削除します。object_name >を、返されたオブジェクトの名前に置き換えます。

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

3. 次のリソースファイルを適用して、Astra Plugin for VMware vSphereサービスがAstra Control Centerサービスに要求を送信できるように、accネットワーキングポリシーオブジェクトを設定します。角かっこ内の情報を環境内の情報に置き換えます。

```
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. 次のリソースファイルを適用して、Astra Control CenterオペレータがAstra Control Centerサービスと通信できるように、acc-operator-network-policyオブジェクトを設定します。角かっこ内の情報を環境内の情報に置き換えます。

```
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
```

リソースの制限を解除します

一部の環境では、ResourceQuotasオブジェクトとLimitRangesオブジェクトを使用して、ネームスペース内のリソースがクラスタ上の使用可能なCPUとメモリをすべて消費しないようにします。Astra Control Centerでは上限が設定されていないため、これらのリソースに準拠していません。Astra Control Centerをインストールするネームスペースから削除する必要があります。

これらのクォータと制限を取得および削除するには、次の手順を実行します。これらの例では、コマンド出力はコマンド出力の直後に表示されます。

手順

1. NetApp-accネームスペース内のリソースクォータを取得します。

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

対応：

```
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. 名前別にすべてのリソースクォータを削除します。

```
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. NetApp-accネームスペース内の制限範囲を取得します。

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

対応：

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

4. 制限範囲を名前で削除します。

```
kubectl delete limitrange cpu-limit-range -n netapp-acc
```

=
:allow-uri-read: