



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	1
Astra Control Centerをダウンロードして展開します	2
ネットアップAstra kubectrlプラグインをインストール	3
イメージをローカルレジストリに追加します	3
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	6
Astra Control Center オペレータを設置します	7
Astra Control Center を設定します	10
Astra Control Center とオペレータのインストールを完了します	25
システムステータスを確認します	26
ロードバランシング用の入力を設定します	31
Astra Control Center UI にログインします	35
インストールのトラブルシューティングを行います	35
次のステップ	36
外部証明書マネージャを設定します	36

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。

その他のインストール手順

- * RedHat OpenShift OperatorHub *でのインストール：これを使用してください ["代替手順"](#) OperatorHubを使用してOpenShiftにAstra Control Centerをインストールするには、次の手順を実行します。
- * Cloud Volumes ONTAP バックエンドを使用してパブリッククラウドにインストール*：ユース ["これらの手順に従います"](#) Amazon Web Services (AWS)、Google Cloud Platform (GCP)、またはCloud Volumes ONTAP ストレージバックエンドを使用するMicrosoft AzureにAstra Control Centerをインストールするには、次の手順を実行します。

Astra Control Centerのインストールプロセスのデモについては、を参照してください ["このビデオでは"](#)。

作業を開始する前に

- ["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 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。
- クラスタに証明書マネージャがすでに存在する場合は、いくつかの手順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は独自の証明書マネージャのインストールを試みません。デフォルトでは、Astra Control Centerはインストール時に独自の証明書マネージャをインストールします。



3つ目の障害ドメインまたはセカンダリサイトにAstra Control Centerを導入これは、アプリケーションのレプリケーションとシームレスなディザスタリカバリに推奨されます。

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、次の作業を行うことができます。

- にAstraコンポーネントを取り付けます netapp-acc (またはカスタム名) ネームスペース。
- デフォルトのAstra Control Owner管理者アカウントを作成します。

- 管理ユーザのEメールアドレスとデフォルトの初期セットアップ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稼働していることを確認します。
- Astra Control Center UIをインストールします。



Astra Control Centerオペレータ (たとえば、`kubectl delete -f astra_control_center_operator_deploy.yaml`) Astra Control Centerのインストール中または操作中はいつでも、ポッドを削除しないようにします。

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をダウンロードして展開します](#)
- [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をダウンロードして展開します

1. にアクセスします ["Astra Control Centerのダウンロードページ"](#) をクリックしますNetApp Support Site。
2. Astra Control Centerを含むバンドルをダウンロードします (`astra-control-center-[version].tar.gz`)。
3. (推奨ですがオプション) Astra Control Centerの証明書と署名のバンドルをダウンロードします (`astra-control-center-certs-[version].tar.gz`) バンドルの署名を確認するには、次の手順を実行します。

```
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
```

出力にはと表示されます Verified OK 検証が成功したあとに、

4. Astra Control Centerバンドルからイメージを抽出します。

```
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

NetApp Astra kubectlコマンドラインプラグインを使用して、ローカルのDockerリポジトリにイメージをプッシュできます。

作業を開始する前に

ネットアップでは、CPUアーキテクチャやオペレーティングシステム別にプラグインのバイナリを提供しています。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。

以前のインストールからプラグインがインストールされている場合は、["最新バージョンがインストールされていることを確認してください"](#) これらの手順を実行する前に。

手順

1. 使用可能なNetApp Astra kubectlプラグインのバイナリを表示し、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。



kubectlプラグインライブラリはtarバンドルの一部であり、フォルダに解凍されます kubectl-astra。

```
ls kubectl-astra/
```

2. 正しいバイナリを現在のパスに移動し、名前をに変更します kubectl-astra :

```
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. tarballのルートディレクトリに移動します。次のファイルとディレクトリが表示されます。

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

2. Astra Control Centerのイメージディレクトリにある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。を実行する前に、次の置換を行ってください push-images コマンドを実行します
 - <BUNDLE_FILE> をAstra Controlバンドルファイルの名前に置き換えます (acc.manifest.bundle.yaml)。
 - <MY_FULL_REGISTRY_PATH> をDockerリポジトリのURLに置き換えます。次に例を示します。 "<a href="https://<docker-registry>" class="bare">https://<docker-registry>"。
 - <MY_REGISTRY_USER> をユーザ名に置き換えます。
 - <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_FULL_REGISTRY_PATH> -u <MY_REGISTRY_USER> -p
<MY_REGISTRY_TOKEN>
```

ポドマン

1. tarballのルートディレクトリに移動します。次のファイルとディレクトリが表示されます。

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

2. レジストリにログインします。

```
podman login <YOUR_REGISTRY>
```

3. 使用するPodmanのバージョンに合わせてカスタマイズされた次のいずれかのスクリプトを準備して実行します。<MY_FULL_REGISTRY_PATH> を'サブディレクトリを含むリポジトリのURLに置き換えます

```
<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
```



レジストリ設定に応じて、スクリプトが作成するイメージパスは次のようになります。

<https://netappdownloads.jfrog.io/docker-astra-control-prod/netapp/astra/acc/23.04.2-7/image:version>

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]
```



インストールを完了する前に、KUBECONFIGがAstra Control Centerをインストールするクラスタを指していることを確認してください。KUBECONFIGには、1つのコンテキストのみを含めることができます。

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。



ブレースホルダ `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/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]
```

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した場合は、ネームスペースを再作成し、ネームスペースのシークレットを再生成します。

- c. を作成します netapp-acc（またはカスタム名）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（またはカスタム名）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
```

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](#)。

```
<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. 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.yaml)
アカウント、サポート、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。

```
vim astra_control_center.yaml
```



注釈付きサンプルYAMLは以下の手順に従います。

2. 次の設定を変更または確認します。

`<code>accountName</code>`

設定	ガイダンス (Guidance)	を入力します	例
accountName	を変更します accountName stringには、Astra Control Centerアカウントに関連付ける名前を指定します。アカウント名は1つだけです。	文字列	Example

`<code>astraVersion</code>`

設定	ガイダンス (Guidance)	を入力します	例
astraVersion	導入するAstra Control Centerのバージョン。この設定には値があらかじめ入力されているため、対処は不要です。	文字列	23.04.2-7

`<code>astraAddress</code>`

設定	ガイダンス (Guidance)	を入力します	例
astraAddress	<p>を変更します</p> <p>astraAddress ブラウザで使用するFQDN (推奨) またはIPアドレスを指定して、Astra Control Centerにアクセスします。このアドレスは、データセンターでAstra Control Centerがどのように検出されるかを定義します。このアドレスは、完了時にロードバランサからプロビジョニングしたFQDNまたはIPアドレスと同じです "Astra Control Center の要件"。</p> <p>注：は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。</p>	文字列	astra.example.com

<code>autoSupport</code>

このセクションで選択することで、ネットアップのプロアクティブサポートアプリケーション、NetApp Active IQ、およびデータの送信先のどちらに参加するかが決まります。インターネット接続が必要です（ポート442）。サポートデータはすべて匿名化されます。

設定	使用	ガイダンス (Guidance)	を入力します	例
<code>autoSupport.enrolled</code>	または <code>enrolled</code> または <code>url</code> フィールドを選択する必要があります	変更 <code>enrolled</code> を選択します <code>AutoSupport false</code> 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 <code>true</code> 接続されているサイト 用の設定 <code>true</code> サポート目的で匿名データをNetAppに送信できるようにします。デフォルトの選択は <code>false</code> およびは、サポートデータがネットアップに送信されないことを示します。	ブール値	<code>false</code> (デフォルト値)
<code>autoSupport.url</code>	または <code>enrolled</code> または <code>url</code> フィールドを選択する必要があります	このURLは匿名データの送信先を決定します。	文字列	https://support.netapp.com/asupprod/post/1.0/postAsup

<code>email</code>

設定	ガイダンス (Guidance)	を入力します	例
email	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 。この E メールアドレスは、最初のアカウントが UI にログインする際のユーザ名として使用され、Astra Control のイベントが通知されます。	文字列	admin@example.com

<code>firstName</code>

設定	ガイダンス (Guidance)	を入力します	例
firstName	アストラアカウントに関連付けられている初期管理者の名前。ここで使用した名前は、初回ログイン後に UI の見出しに表示されます。	文字列	SRE

<code>LastName</code>

設定	ガイダンス (Guidance)	を入力します	例
lastName	アストラアカウントに関連付けられている初期管理者の姓です。ここで使用した名前は、初回ログイン後に UI の見出しに表示されます。	文字列	Admin

<code>imageRegistry</code>

このセクションで選択すると、Astraアプリケーションイメージ、Astra Control Center Operator、Astra Control Center Helmリポジトリをホストするコンテナイメージレジストリが定義されます。

設定	使用	ガイダンス (Guidance)	を入力します	例
<code>imageRegistry.name</code>	必須	でイメージをプッシュしたイメージレジストリの名前の手順。使用しないでください http:// または https:// をレジストリ名に追加します。	文字列	<code>example.registry.com/astra</code>
<code>imageRegistry.secret</code>	に入力した文字列の場合は必須です <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> ラインの内側 <code>imageRegistry</code> または、インストールが失敗します。	イメージレジストリでの認証に使用するKubernetesシークレットの名前。	文字列	<code>astra-registry-cred</code>

<code>storageClass</code>

設定	ガイダンス (Guidance)	を入力します	例
storageClass	<p>を変更します</p> <p>storageClass からの値 <code>ontap-gold</code> インストール環境で必要な別の Astra Trident storageClass リソースに移動します。コマンドを実行します</p> <p><code>kubectl get sc</code> をクリックして、設定済みの既存のストレージクラスを確認します。Astra Trident ベースのストレージクラスのいずれかをマニフェストファイルに入力する必要があります</p> <p>(<code>astra-control-center- <version>.manifest</code>) とを Astra PVS に使用します。設定されていない場合は、デフォルトのストレージクラスが使用されます。</p> <p>メモ：デフォルトのストレージクラスが設定されている場合は、デフォルトのアノテーションが設定されている唯一のストレージクラスであることを確認してください。</p>	文字列	<code>ontap-gold</code>

`<code>volumeReclaimPolicy</code>`

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
volumeReclaimPolicy	これにより、AstraのPVSの再利用ポリシーが設定されます。このポリシーをに設定しています Retain Astraが削除されたあとに永続的なボリュームを保持このポリシーをに設定しています Delete Astraが削除されたあとに永続的ボリュームを削除する。この値が設定されていない場合、PVSは保持されま	文字列	<ul style="list-style-type: none">• Retain (デフォルト値)• Delete

`<code>ingressType</code>`





設定	ガイダンス (Guidance)	を入力します	オプション (Options)
ingressType	<p>次の入力タイプのいずれかを使用します。</p> <p>Generic (ingressType: "Generic") (デフォルト) このオプションは、別の入力コントローラを使用している場合、または独自の入力コントローラを使用する場合に使用します。Astra Control Centerを導入したら、を設定する必要があります "入力コントローラ" URLを使用してAstra Control Centerを公開します。</p> <p>AccTraefik (ingressType: "AccTraefik") 入力コントローラを設定しない場合は、このオプションを使用します。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。</p> <p>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</p>	文字列	<ul style="list-style-type: none"> • Generic (デフォルト値) • AccTraefik

<code>scaleSize</code>

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
scaleSize	<p>デフォルトでは、Astraで高可用性 (HA) が使用されます。</p> <p>scaleSize の Medium`ほとんどのサービスをHAに導入し、冗長性を確保するために複数のレプリカを導入します。を使用`scaleSizeとして Small`Astraは、消費量を削減するための必須サービスを除き、すべてのサービスのレプリカ数を削減します。</p> <p>ヒント： `Medium`環境は約100個のポッドで構成されています (一時的なワークロードは含まれません)。100個のポッドは、3つのマスターノードと3つのワーカーノード構成に基づいています)。特にディザスタリカバリのシナリオを検討する場合は、環境で問題となる可能性があるポッド単位のネットワーク制限に注意してください。</p>	文字列	<ul style="list-style-type: none">• Small• Medium (デフォルト値)

<code>astraResourcesScaler</code>

設定	ガイダンス (Guidance)	を入力します	オプション (Options)
astraResourcesScaler	<p>AstraeControlCenterリソース制限のスケールリングオプションデフォルトでは、Astra Control CenterはAstra内のほとんどのコンポーネントに対してリソース要求を設定して展開します。この構成により、アプリケーションの負荷と拡張性が高い環境では、Astra Control Centerソフトウェアスタックのパフォーマンスが向上します。</p> <p>ただし、小規模な開発またはテストクラスタを使用するシナリオでは、CRフィールドを使用します</p> <p>astraResourcesScaler に設定できます Off。これにより、リソース要求が無効になり、小規模なクラスタへの導入が可能になります。</p>	文字列	<ul style="list-style-type: none">• Default (デフォルト値)• Off

<code>additionalValues</code>

- アストラルコントロールセンターおよびCloud Insights 通信では、TLS証明書の検証はデフォルトで無効になっています。の次のセクションを追加して、Cloud Insights とAstra Control Center のホストクラスタと管理対象クラスタの両方の間の通信に対してTLS証明書の検証を有効にすることができます additionalValues。

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

<code>crds</code>

このセクションで選択した内容によって、Astra Control CenterでのCRDの処理方法が決まります。

設定	ガイダンス (Guidance)	を入力します	例
<code>crds.externalCertManager</code>	<p>外部証明書マネージャを使用する場合は、変更します</p> <p><code>externalCertManager</code> 終了: <code>true</code>。デフォルト <code>false</code> Astra Control Centerが、インストール時に独自の証明書マネージャCRDをインストールするようにします。</p> <p>SSDはクラスタ全体のオブジェクトであり、クラスタの他の部分に影響を及ぼす可能性があります。このフラグを使用すると、これらのCRDがAstra Control Centerの外部にあるクラスタ管理者によってインストールおよび管理されることをAstra Control Centerに伝えることができます。</p>	ブール値	False (デフォルト値)
<code>crds.externalTraefik</code>	<p>デフォルトでは、Astra Control Centerは必要なTraefik CRDをインストールします。SSDはクラスタ全体のオブジェクトであり、クラスタの他の部分に影響を及ぼす可能性があります。このフラグを使用すると、これらのCRDがAstra Control Centerの外部にあるクラスタ管理者によってインストールおよび管理されることをAstra Control Centerに伝えることができます。</p>	ブール値	False (デフォルト値)



インストールを完了する前に、構成に適したストレージクラスと入力タイプを選択していることを確認してください。

```
<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
```

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）名前スペース：

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

回答例:

```
astracenter.astra.netapp.io/astra created
```



Astra Control Centerのオペレータが環境要件の自動チェックを実行ありません "要件" 原因でインストールが失敗するか、Astra Control Centerが正常に動作しない可能性があります。を参照してください [次のセクション](#) 自動システムチェックに関連する警告メッセージをチェックします。

システムステータスを確認します

kubectlコマンドを使用すると、システムステータスを確認できます。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。

手順

1. インストールプロセスで検証チェックに関連する警告メッセージが生成されなかったことを確認します。

```
kubectl get acc [astra or custom Astra Control Center CR name] -n [netapp-acc or custom namespace] -o yaml
```



その他の警告メッセージは、Astra Control Centerのオペレータログでも報告されます。

2. 自動化された要件チェックによって報告された環境の問題を修正します。



問題を解決するには、環境が満たしていることを確認します "要件" (Astra Control Center向け)。

3. 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ります。

回答例

NAME	READY	STATUS	
RESTARTS AGE			
acc-helm-repo-6cc7696d8f-pmhm8 9h	1/1	Running	0
activity-597fb656dc-5rd4l 9h	1/1	Running	0
activity-597fb656dc-mqmcw 9h	1/1	Running	0
api-token-authentication-62f84 9h	1/1	Running	0
api-token-authentication-68nlf 9h	1/1	Running	0
api-token-authentication-ztgrm 9h	1/1	Running	0
asup-669d4ddbc4-fnmwp (9h ago) 9h	1/1	Running	1
authentication-78789d7549-1k686 9h	1/1	Running	0
bucket-service-65c7d95496-24x7l (9h ago) 9h	1/1	Running	3
cert-manager-c9f9fbf9f-k8zq2 9h	1/1	Running	0
cert-manager-c9f9fbf9f-qjllzm 9h	1/1	Running	0
cert-manager-cainjector-dbbbd8447-b5ql1 9h	1/1	Running	0
cert-manager-cainjector-dbbbd8447-p5whs 9h	1/1	Running	0
cert-manager-webhook-6f97bb7d84-4722b 9h	1/1	Running	0
cert-manager-webhook-6f97bb7d84-86kv5 9h	1/1	Running	0
certificates-59d9f6f4bd-2j899 9h	1/1	Running	0
certificates-59d9f6f4bd-9d9k6 9h	1/1	Running	0
certificates-expiry-check-28011180--1-8lkxz 9h	0/1	Completed	0
cloud-extension-5c9c9958f8-jdhrp 9h	1/1	Running	0
cloud-insights-service-5cdd5f7f-pp8r5 9h	1/1	Running	0
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. (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記録されている場合は、を使用して再度登録を試行できます ["UIでクラスタワークフローを追加します"](#) または API。

5. 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 `True`) を使用して、Astra Control Centerにログインするときに使用する初期セットアップパスワードを取得します。

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

対応：

NAME	UUID	VERSION	ADDRESS
READY			
astra	9aa5fdae-4214-4cb7-9976-5d8b4c0ce27f	23.04.2-7	10.111.111.111
True			



UUIDの値をコピーします。パスワードは `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

サービスへの外部アクセスを管理するKubernetes入力コントローラを設定できます。これらの手順では、デフォルトのを使用した場合の入力コントローラの設定例を示します `ingressType: "Generic"` Astra Control Centerのカスタムリソース (`astra_control_center.yaml`) 。を指定した場合、この手順を使用する必要はありません `ingressType: "AccTraefik"` Astra Control Centerのカスタムリソース (`astra_control_center.yaml`) 。

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。

セットアップ手順は、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。Astra Control Centerは、多くの入力コントローラタイプをサポートしています。これらのセットアップ手順では、次の入力コントローラタイプの手順の例を示します。

- Istio入力
- nginx 入力コントローラ

- OpenShift 入力コントローラ

作業を開始する前に

- 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。

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. に入力リソースを配置します `netapp-acc` (またはカスタムネームスペース)。スキーマにはv1リソースタイプを使用します (`istio-Ingress.yaml` は次の例で使用されています)。

```

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. 変更を適用します。

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

6. 入力ステータスを確認します。

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

対応:

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

7. Astra Control Centerのインストールを完了します。

Ngix 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` でTLSの秘密鍵と証明書を使用する場合 `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` (またはカスタムネームスペース)。スキーマにはv1リソースタイプを使用します (`nginx-Ingress.yaml` は次の例で使用されています)。

```
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. 変更を適用します。

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



ネットアップでは、nginxコントローラをではなく導入環境としてインストールすることを推奨します `daemonSet`。

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、（を含む）FQDNを入力します `https://` プレフィックス）を使用します `astraAddress` を参照してください `astra_control_center.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、自己署名証明書を承認します。



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.yaml` CR When（時間） [Astra Control Center をインストールした](#) をクリックし、次に初期セットアップ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。



初めてログインしたときにパスワードを忘れ、他の管理ユーザアカウントがまだ作成されていない場合は、にお問い合わせください ["ネットアップサポート"](#) パスワード回復のサポートを受けるには、

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。

オプション（Options）

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

- Astra Control Center CRの出力を確認するには、次の手順を実行します。

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

次のステップ

- (オプション) お使いの環境に応じて、インストール後に実行します "設定手順"。
- を実行して導入を完了します "セットアップのタスク"。

外部証明書マネージャを設定します

Kubernetesクラスタに証明書マネージャがすでに存在する場合は、Astra Control Centerで独自の証明書マネージャがインストールされないように、いくつかの前提条件となる手順を実行する必要があります。

手順

1. 証明書マネージャがインストールされていることを確認します。

```
kubectl get pods -A | grep 'cert-manager'
```

回答例：

```
cert-manager   essential-cert-manager-84446f49d5-sf2zd   1/1
Running        0      6d5h
cert-manager   essential-cert-manager-cainjector-66dc99cc56-91dmt   1/1
Running        0      6d5h
cert-manager   essential-cert-manager-webhook-56b76db9cc-fjqrq     1/1
Running        0      6d5h
```

2. の証明書とキーのペアを作成します astraAddress FQDN：

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

回答例：

```
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'tls.key'
```

3. 以前に生成したファイルを使用してシークレットを作成します。

```
kubectl create secret tls selfsigned-tls --key tls.key --cert tls.crt -n
<cert-manager-namespace>
```

回答例：

```
secret/selfsigned-tls created
```

4. を作成します ClusterIssuer *とまったく同じ*のファイル。ただし、の名前空間の場所が含まれます cert-manager ポッドがインストールされます。

```
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
```

回答例：

```
clusterissuer.cert-manager.io/astra-ca-clusterissuer created
```

5. を確認します ClusterIssuer が正常に起動しました。Ready はである必要があります True 次の手順に進む前に、次の手順

```
kubectl get ClusterIssuer
```

回答例：

NAME	READY	AGE
astra-ca-clusterissuer	True	9s

6. を実行します ["Astra Control Center のインストールプロセス"](#)。があります ["Astra Control Center クラスターYAMLの必須の設定手順"](#) CRD値を変更して、証明書マネージャが外部にインストールされていることを示します。Astra Control Centerが外部証明書マネージャを認識するように、インストール時にこの手順を完了する必要があります。

著作権に関する情報

Copyright © 2023 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および/または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用権を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用権については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。