Astra Control Center をセットアップします

Astra Control Center

NetApp
November 21, 2023

This PDF was generated from https://docs.netapp.com/ja-jp/astra-control-center-2112/get-started/add-cluster-reqs.html on November 21, 2023. Always check docs.netapp.com for the latest.

目次

Astra Control Center をセットアップします . . . 1
- Astra Control Center のライセンスを追加します . . . 1
- クラスタを追加 . . . 2
- ストレージバックエンドを追加します . . . 4
- バケットを追加します . . . 5
- 次の手順 . . . 6
- クラスタを追加するための前提条件 . . . 7
- カスタム TLS 証明書を追加します . . . 12

Astra Control Center をセットアップします

Astra Control Center は、 ONTAP と Astra データストアをストレージバックエンドとしてサポートおよび監視します。Astra Control Center をインストールして UI にログインし、パスワードを変更したら、ライセンスの設定、クラスタの追加、ストレージの管理、バケットの追加を行います。

タスク

- Astra Control Center のライセンスを追加します
- [クラスタを追加]
- [ストレージバックエンドを追加します]
- [バケットを追加します]

Astra Control Center のライセンスを追加します

UI またはを使用して新しいライセンスを追加できます "API" Astra Control Center の全機能を利用できます。ライセンスがないと、 Astra Control Center の使用は、ユーザの管理と新しいクラスタの追加に限定されます。

必要なもの

から Astra Control Center をダウンロードした場合 "ネットアップサポートサイト"には、 NetApp License File （ NLF ）もダウンロードします。このライセンスファイルにアクセスできることを確認してください。

> 既存の評価版またはフルライセンスを更新するには、を参照してください "既存のライセンスを更新する"。

フルライセンスまたは評価ライセンスを追加します

Astra Control Center ライセンスは、 Kubernetes CPU ユニットを使用して CPU リソースを測定します。ライセンスには、管理対象のすべての Kubernetes クラスタのワーカーノードに割り当てられた CPU リソースが含まれている必要があります。ライセンスを追加する前に、からライセンスファイル（ NLF ）を取得する必要があります "ネットアップサポートサイト"。

また、 Astra Control Center に評価ライセンスをお試しいただくこともできます。このライセンスは、 Astra Control Center をダウンロードした日から 90 日間使用できます。登録すると、無償トライアルに登録できます "こちらをご覧ください"。

> インストールがライセンス数を超えると、 Astra Control Center は新しいアプリケーションを管理できなくなります。容量を超えるとアラートが表示されます。

手順

1. Astra Control Center UI にログインします。
2. 「 * アカウント * > * ライセンス * 」を選択します。
3. 「 * ライセンスの追加 * 」を選択します。
4. ダウンロードしたライセンスファイル（ NLF ）を参照します。
5. 「 * ライセンスの追加 * 」を選択します。

Account>*License* ページには、ライセンス情報、有効期限、ライセンスシリアル番号、アカウント ID 、および使用されている CPU ユニットが表示されます。

評価用ライセンスをお持ちの場合は、 Astra Control Center に障害が発生したときに ASUP を送信していないときにデータが失われないように、アカウント ID を必ず保存してください。

クラスタを追加

アプリケーションの管理を開始するには、 Kubernetes クラスタを追加し、コンピューティングリソースとして管理します。Kubernetes アプリケーションを検出するには、 Astra Control Center のクラスタを追加する必要があります。Astra データストアプレビューでは、 Astra データストアプレビューでプロビジョニングしたボリュームを使用するアプリケーションを含む Kubernetes アプリケーションクラスタを追加します。

他のクラスタを Astra Control Center に追加して管理する前に、 Astra Control Center が最初に導入したクラスタを管理することをお勧めします。指標およびトラブルシューティング用の Kubemetrics データとクラスタ関連データを送信するには、最初のクラスタを管理下に配置する必要があります。Add Cluster * 機能を使用して、 Astra Control Center でクラスタを管理できます。

Astra Control は、クラスタを管理するときに、クラスタのデフォルトのストレージクラスを追跡します。'kubectl' コマンドを使用して StorageClass を変更すると 'Astra Control は変更を元に戻しますAstra Control によって管理されるクラスタのデフォルトの StorageClass を変更するには、次のいずれかの方法を使用します。

- Astra Control API’put/managedClusters' エンドポイントを使用して 'DefaultStorageClass' パラメータを指定して ' 別のデフォルト StorageClass を割り当てます
- Astra Control 管理からクラスタを削除し、別のデフォルトの StorageClass を選択して追加し直します

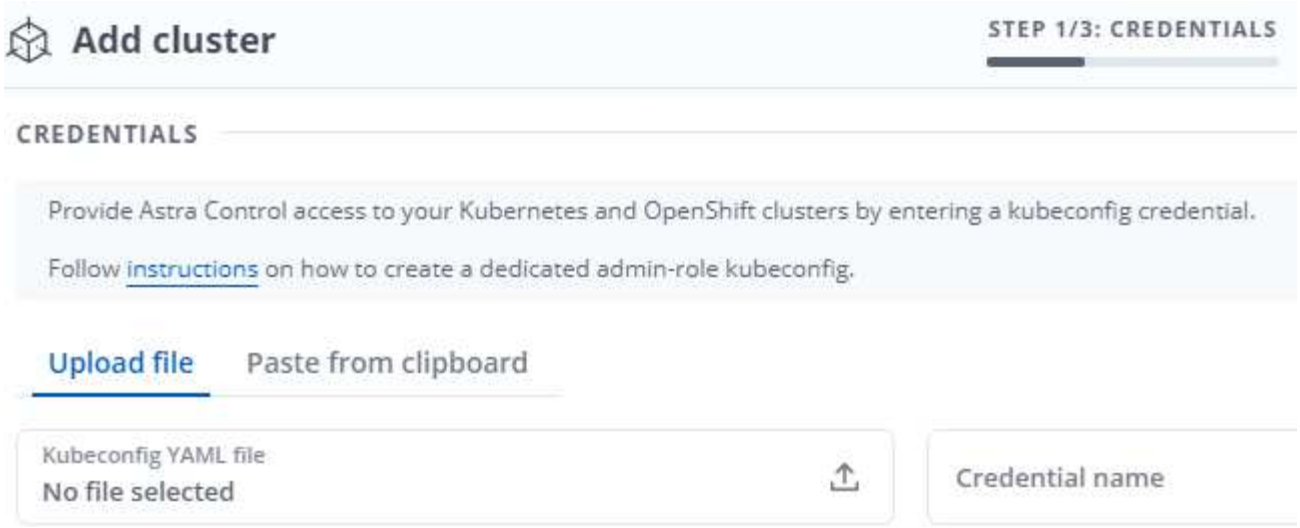
必要なもの

クラスタを追加する前に、必要なを確認し、実行しておきます "前提条件となるタスク"。

手順

1. Astra Control Center UI の * ダッシュボード * から、クラスタセクションで * 追加 * を選択します。
2. 表示された「クラスタを追加」ウィンドウで、「 kubeconfig .yaml 」ファイルをアップロードするか、「 kubeconfig .yaml 」ファイルの内容を貼り付けます。

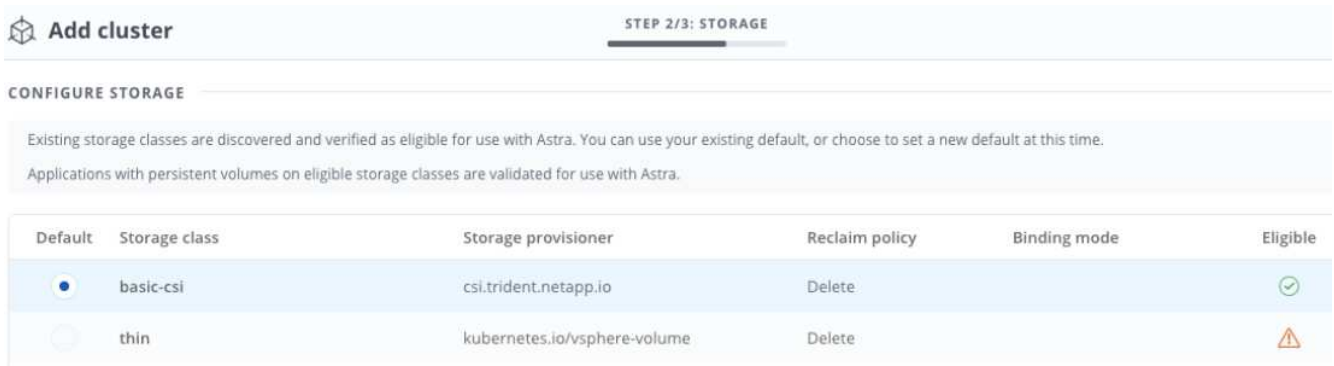
「 kubeconfig .yaml 」ファイルには、 1 つのクラスタのクラスタ・クレデンシャルのみを含める必要があります。



独自の「 kubeconfig 」ファイルを作成する場合は、その中で * 1 つの * コンテキストエレメントのみを定義する必要があります。を参照してください "Kubernetes のドキュメント" 「 kubeconfig 」ファイルの作成方法については、を参照してください。

3. クレデンシャル名を指定します。デフォルトでは、クレデンシャル名がクラスタの名前として自動的に入力されます。
4. 「ストレージの設定」を選択します。
5. この Kubernetes クラスタに使用するストレージクラスを選択し、 * Review * を選択します。

ONTAP ストレージまたは Astra データストアからバックアップされた Trident ストレージクラスを選択する必要があります。



6. 情報を確認し、問題がない場合は「 * クラスタの追加 * 」を選択します。

結果

クラスタが「 Discovering * 」ステータスになり、「 Running 」に変わります。Kubernetes クラスタが正常に追加され、 Astra Control Center で管理できるようになりました。

Astra Control Center で管理するクラスタを追加したあと、監視オペレータの配置に数分かかる場合があります。それまでは、通知アイコンが赤に変わり、 * モニタリングエージェントステータスチェック失敗 * イベントが記録されます。この問題は無視してかまいません。問題は、Astra Control Center が正しいステータスを取得したときに解決します。数分で問題が解決しない場合は、クラスタに移動し、「 OC get pod -n NetApp-monitoring 」を開始点として実行します。問題をデバッグするには、監視オペレータのログを調べる必要があります。

ストレージバックエンドを追加します

ストレージバックエンドを追加して、 Astra Control がリソースを管理できるようにすることができます。ストレージバックエンドとして Astra Control のストレージクラスタを管理することで、永続ボリューム（ PVS ）とストレージバックエンドの間のリンケージを取得できるだけでなく、追加のストレージ指標も取得できます。

検出されたストレージバックエンドを追加するには、ダッシュボードまたはバックエンドメニューからプロンプトを選択します。

必要なもの

- これで完了です "クラスタが追加されました" また、 Astra Control によって管理されます。

  管理対象クラスタには、 Astra Control によって検出可能なサポート対象のバックエンドが接続されています。

- Astra データストアプレビューインストールの場合： Kubernetes アプリケーションクラスタを追加しておきます。

  Astra Data Store 用の Kubernetes アプリケーションクラスタを追加すると、検出されたバックエンドのリストに「 unmanaged 」と表示されます。次に、 Astra データストアを含むコンピューティングクラスタを追加し、 Kubernetes アプリケーションクラスタの基盤を構築する必要があります。これは、 UI の * Backends * から実行できます。クラスタの [Actions] メニューを選択し、 [`Manage] を選択して、およびを選択します "クラスタを追加"。「 unmanaged 」のクラスタ状態が Kubernetes クラスタの名前に変わったら、バックエンドの追加に進むことができます。

手順

1. 次のいずれかを実行します。
   - ダッシュボードから * ：
     i. ダッシュボードストレージバックエンドセクションで、 * 管理 * を選択します。
     ii. Dashboard Resource Summary > Storage Backends セクションで、 * Add * を選択します。
   - バックエンドから * ：
     i. 左側のナビゲーション領域で、 * Backends * を選択します。
     ii. 「 * Manage * 」を選択します。
2. バックエンドの種類に応じて、次のいずれかの操作を行います。
   - * Astra データストア * ：

i. 「 * Astra Data Store * 」タブを選択します。

ii. 管理対象のコンピューティングクラスタを選択し、 * Next * を選択します。

iii. バックエンドの詳細を確認し、「 * ストレージバックエンドの管理 * 」を選択します。

◦ * ONTAP * ：

i. ONTAP の管理者クレデンシャルを入力し、「 * Review * 」を選択します。

ii. バックエンドの詳細を確認し、 * Manage * を選択します。

バックエンドは ' サマリー情報とともに ' リスト内の [Available （使用可能）] 状態で表示されます

バックエンドが表示されるようにページを更新する必要がある場合があります。

バケットを追加します

アプリケーションと永続的ストレージをバックアップする場合や、クラスタ間でアプリケーションのクローニングを行う場合は、オブジェクトストアバケットプロバイダの追加が不可欠です。Astra Control は、これらのバックアップまたはクローンを、定義したオブジェクトストアバケットに格納します。

バケットを追加すると、 Astra Control によって、 1 つのバケットがデフォルトのバケットインジケータとしてマークされます。最初に作成したバケットがデフォルトバケットになります。

アプリケーション構成と永続的ストレージを同じクラスタにクローニングする場合、バケットは必要ありません。

次のいずれかのバケットタイプを使用します。

- NetApp ONTAP S3
- NetApp StorageGRID S3 の略
- 汎用 S3

Astra Control Center は Amazon S3 を汎用 S3 バケットプロバイダとしてサポートしていますが、 Astra Control Center は Amazon の S3 サポートを要求するすべてのオブジェクトストアベンダーをサポートしているわけではありません。

Astra Control API を使用してバケットを追加する手順については、を参照してください "Astra の自動化と API に関する情報"。

手順

1. 左側のナビゲーション領域で、 * バケット * を選択します。
   a. 「 * 追加」を選択します。
   b. バケットタイプを選択します。

バケットを追加するときは、正しいバケットプロバイダを選択し、そのプロバイダに適したクレデンシャルを指定します。たとえば、タイプとして NetApp ONTAP S3 が許可され、 StorageGRID クレデンシャルが受け入れられますが、このバケットを使用して原因の以降のアプリケーションのバックアップとリストアはすべて失敗します。

c. 新しいバケット名を作成するか、既存のバケット名とオプションの概要を入力します。

> バケット名と概要は、バックアップを作成するときに後で選択できるバックアップの場所として表示されます。この名前は、保護ポリシーの設定時にも表示されます。

d. S3 エンドポイントの名前または IP アドレスを入力します。

e. このバケットをすべてのバックアップのデフォルトバケットにする場合は、「このバケットをこのプライベートクラウドのデフォルトバケットにする」オプションを選択します。

> このオプションは、最初に作成したバケットに対しては表示されません。

f. 追加して続行します クレデンシャル情報。

S3 アクセスクレデンシャルを追加します

S3 アクセスクレデンシャルはいつでも追加できます。

手順

1. バケット（ Buckets ）ダイアログで、 * 追加（ Add ） * または * 既存の * を使用（ Use Existing * ）タブのいずれかを選択します。
   a. Astra Control の他のクレデンシャルと区別するクレデンシャルの名前を入力します。
   b. クリップボードからコンテンツを貼り付けて、アクセス ID とシークレットキーを入力します。

次の手順

Astra Control Center にログインしてクラスタを追加したので、 Astra Control Center のアプリケーションデータ管理機能を使い始めることができます。

- "ユーザを管理します"
- "アプリの管理を開始します"
- "アプリを保護します"
- "アプリケーションをクローニング"
- "通知を管理します"
- "Cloud Insights に接続します"
- "カスタム TLS 証明書を追加します"

詳細については、こちらをご覧ください

- "Astra Control API を使用"
- "既知の問題"

クラスタを追加するための前提条件

クラスタを追加する前に、前提条件が満たされていることを確認する必要があります。また、資格チェックを実行して、アストラコントロールセンターにクラスタを追加する準備ができていることを確認する必要があります。

クラスタを追加する前に必要な作業

- 次のいずれかのタイプのクラスタ：
  - OpenShift 4.6 、 4.7 、または 4.8 を実行するクラスタ。 Astra Trident StorageClasses を搭載し、Astra Data Store または ONTAP 9.5 以降をサポートしています
  - Rancher 2.5 を実行しているクラスタ
  - Kubernetes 1.19 ~ 1.21 を実行するクラスタ（ 1.21x を含む）

    クラスタに 1 つ以上のワーカーノードがあり、テレメトリサービスの実行には 1GB 以上の RAM が使用されていることを確認します。

> 管理対象のコンピューティングリソースとして 2 つ目の OpenShift 4.6 、 4.7 、または 4.8 クラスタを追加する場合は、 Astra Trident ボリュームスナップショット機能が有効になっていることを確認する必要があります。ネットアップの公式 Astra Trident をご覧ください "手順" Trident を使用して、ボリューム Snapshot を有効にしてテストしてください。

- Astra Control Center を使用してアプリケーションをバックアップおよび復元するためにバックアップ ONTAP システムで設定されたスーパーユーザーおよびユーザー ID 。ONTAP コマンドラインで次のコマンドを実行します。「 export-policy rule modify -vserver <storage virtual machine name> -policyname <policy name> -ruleindex 1 -superuser sysm — anon 65534 」
- 管理者によって定義された Astra Trident 'volumesnapshotclass' オブジェクトAstra Trident の詳細をご確認ください "手順" Trident を使用して、ボリューム Snapshot を有効にしてテストしてください。
- Kubernetes クラスタにデフォルトのストレージクラスが 1 つだけ定義されていることを確認します。

資格チェックを実行します

次の資格チェックを実行して、 Astra Control Center にクラスタを追加する準備ができていることを確認します。

手順

1. Trident のバージョンを確認

```
kubectl get tridentversions -n trident
```

Trident が存在する場合は、次のような出力が表示されます。

```
NAME      VERSION
trident   21.04.0
```

Trident が存在しない場合は、次のような出力が表示されます。

```
error: the server doesn't have a resource type "tridentversions"
```

Trident がインストールされていない場合や、インストールされているバージョンが最新でない場合は、次に進む前に最新バージョンの Trident をインストールする必要があります。を参照してください "Trident のドキュメント" 手順については、を参照し

2. サポートされている Trident ドライバをストレージクラスが使用しているかどうかを確認します。プロビジョニング担当者の名前は「 csi.trident.netapp.io` 」にする必要があります。次の例を参照してください。

```
kubectl get sc
NAME                     PROVISIONER                    RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)    csi.trident.netapp.io          Delete
Immediate           true                   5d23h
thin                    kubernetes.io/vsphere-volume   Delete
Immediate           false                  6d
```

admin ロールの kubeconfig を作成します

手順を実行する前に、マシンに次のものがあることを確認してください。

- kubectl’V1.19 以降がインストールされています
- アクティブなコンテキストのクラスタ管理者権限があるアクティブな kubeconfig です

手順

1. 次の手順でサービスアカウントを作成します。

   a. 「 astracontrol-service-account.yaml 」という名前のサービスアカウントファイルを作成します。

      名前と名前空間を必要に応じて調整します。ここで変更を行った場合は、以降の手順でも同じ変更を適用する必要があります。

```
<strong>astracontrol-service-account.yaml</strong>
```

+

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: astracontrol-service-account
  namespace: default
```

a. サービスアカウントを適用します。

```
kubectl apply -f astracontrol-service-account.yaml
```

2. 次のようにクラスタ管理者権限を付与します。

a. 「 astracontrol-clusterrolbinding.yaml 」という名前の「 ClusterRoleBinding 」ファイルを作成します。

必要に応じて、サービスアカウントの作成時に変更した名前と名前空間を調整します。

```
<strong>astracontrol-clusterrolebinding.yaml</strong>
```

+

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: astracontrol-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: astracontrol-service-account
  namespace: default
```

a. クラスタロールバインドを適用します。

```
kubectl apply -f astracontrol-clusterrolebinding.yaml
```

3. 「 <context> 」をインストールの正しいコンテキストに置き換えて、サービスアカウントのシークレットをリストします。

```
kubectl get serviceaccount astracontrol-service-account --context
<context> --namespace default -o json
```

出力の末尾は次のようになります。

```
"secrets": [
{ "name": "astracontrol-service-account-dockercfg-vhz87"},
{ "name": "astracontrol-service-account-token-r59kr"}
]
```

'ecsレット ' 配列内の各要素のインデックスは 0 から始まります上記の例では、「 astracontrol-service-account-dockercfg-vhz87 」のインデックスは 0 になり、「 astracontrol-service-account-token-r59kr 」のインデックスは 1 になります。出力で、 "token" という単語が含まれるサービスアカウント名のインデックスをメモしてください。

4. 次のように kubeconfig を生成します。
   a. 「 create-kubeconfig .sh` ファイル」を作成します。次のスクリプトの先頭にある「 token_index 」を正しい値に置き換えます。

```
<strong>create-kubeconfig.sh</strong>
```

```
# Update these to match your environment.
# Replace TOKEN_INDEX with the correct value
# from the output in the previous step. If you
# didn't change anything else above, don't change
# anything else here.

SERVICE_ACCOUNT_NAME=astracontrol-service-account
NAMESPACE=default
NEW_CONTEXT=astracontrol
KUBECONFIG_FILE='kubeconfig-sa'

CONTEXT=$(kubectl config current-context)

SECRET_NAME=$(kubectl get serviceaccount ${SERVICE_ACCOUNT_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.secrets[TOKEN_INDEX].name}')
TOKEN_DATA=$(kubectl get secret ${SECRET_NAME} \
  --context ${CONTEXT} \
  --namespace ${NAMESPACE} \
  -o jsonpath='{.data.token}')
```

```
TOKEN=$(echo ${TOKEN_DATA} | base64 -d)

# Create dedicated kubeconfig
# Create a full copy
kubectl config view --raw > ${KUBECONFIG_FILE}.full.tmp

# Switch working context to correct context
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp config use-context
${CONTEXT}

# Minify
kubectl --kubeconfig ${KUBECONFIG_FILE}.full.tmp \
  config view --flatten --minify > ${KUBECONFIG_FILE}.tmp

# Rename context
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  rename-context ${CONTEXT} ${NEW_CONTEXT}

# Create token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-credentials ${CONTEXT}-${NAMESPACE}-token-user \
  --token ${TOKEN}

# Set context to use token user
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --user ${CONTEXT}-${NAMESPACE}-token-
user

# Set context to correct namespace
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  set-context ${NEW_CONTEXT} --namespace ${NAMESPACE}

# Flatten/minify kubeconfig
kubectl config --kubeconfig ${KUBECONFIG_FILE}.tmp \
  view --flatten --minify > ${KUBECONFIG_FILE}

# Remove tmp
rm ${KUBECONFIG_FILE}.full.tmp
rm ${KUBECONFIG_FILE}.tmp
```

b. コマンドをソースにし、 Kubernetes クラスタに適用します。

```
source create-kubeconfig.sh
```

5. （ * オプション * ）クラスタにわかりやすい名前にコベ econfig の名前を変更します。クラスタのクレデ

ンシャルを保護します。

```
chmod 700 create-kubeconfig.sh
mv kubeconfig-sa.txt YOUR_CLUSTER_NAME_kubeconfig
```

次の手順

前提条件が満たされていることを確認したら、次は準備ができています "クラスタを追加"。

詳細については、こちらをご覧ください

- "Trident のドキュメント"
- "Astra Control API を使用"

カスタム TLS 証明書を追加します

既存の自己署名 TLS 証明書を削除して、認証局（ CA ）が署名した TLS 証明書に置き換えることができます。

必要なもの

- Astra Control Center をインストールした Kubernetes クラスタ
- クラスタ上のコマンド・シェルに管理アクセスして 'kubectl' コマンドを実行します
- CA の秘密鍵ファイルと証明書ファイル

自己署名証明書を削除します

既存の自己署名 TLS 証明書を削除します。

1. SSH を使用して、 Astra Control Center をホストする Kubernetes クラスタに管理ユーザとしてログインします。
2. 次のコマンドを使用して、現在の証明書に関連付けられている TLS シークレットを検索します。「 <ACC-deployment-namespace> 」は、 Astra Control Center 配置名前空間に置き換えてください。

```
kubectl get certificate -n <ACC-deployment-namespace>
```

3. 次のコマンドを使用して、現在インストールされているシークレットと証明書を削除します。

```
kubectl delete cert cert-manager-certificates -n <ACC-deployment-
namespace>
kubectl delete secret secure-testing-cert -n <ACC-deployment-namespace>
```

新しい証明書を追加します

CA によって署名された新しい TLS 証明書を追加します。

1. 次のコマンドを使用して、 CA の秘密鍵ファイルと証明書ファイルを使用して新しい TLS シークレットを作成し、括弧 <> の引数を適切な情報に置き換えます。

```
kubectl create secret tls <secret-name> --key <private-key-filename>
--cert <certificate-filename> -n <ACC-deployment-namespace>
```

2. 次のコマンドと例を使用して、クラスタカスタムリソース定義（ CRD ）ファイルを編集し、「 pec.selfSigned` 」の値を「 pec.ca.secretName` 」に変更して、前の手順で作成した TLS シークレットを参照します。

```
kubectl edit clusterissuers.cert-manager.io/cert-manager-certificates -n
<ACC-deployment-namespace>
....

#spec:
#  selfSigned: {}

spec:
  ca:
    secretName: <secret-name>
```

3. 次のコマンドと出力例を使用して、変更が正しく、クラスタが証明書を検証できる状態にあることを確認します。「 <ACC-deployment-namespace> 」は Astra Control Center 配置名前空間に置き換えてください。

```
kubectl describe clusterissuers.cert-manager.io/cert-manager-
certificates -n <ACC-deployment-namespace>
....

Status:
  Conditions:
    Last Transition Time:  2021-07-01T23:50:27Z
    Message:               Signing CA verified
    Reason:                KeyPairVerified
    Status:                True
    Type:                  Ready
Events:                    <none>
```

4. 次の例を使用して 'certificate.yaml ファイルを作成します括弧 <> のプレースホルダ値を適切な情報に置き換えます

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: <certificate-name>
  namespace: <ACC-deployment-namespace>
spec:
  secretName: <certificate-secret-name>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  - <astra.dnsname.example.com> #Replace with the correct Astra Control
Center DNS address
  issuerRef:
    kind: ClusterIssuer
    name: cert-manager-certificates
```

5. 次のコマンドを使用して証明書を作成します。

```
kubectl apply -f certificate.yaml
```

6. 次のコマンドと出力例を使用して、証明書が正しく作成されていること、および作成時に指定した引数（名前、期間、更新期限、 DNS 名など）を使用していることを確認します。

```
kubectl describe certificate -n <ACC-deployment-namespace>
....

Spec:
  Dns Names:
    astra.example.com
  Duration:  125h0m0s
  Issuer Ref:
    Kind:        ClusterIssuer
    Name:        cert-manager-certificates
  Renew Before:  61h0m0s
  Secret Name:   <certificate-secret-name>
Status:
  Conditions:
    Last Transition Time:  2021-07-02T00:45:41Z
    Message:               Certificate is up to date and has not expired
    Reason:                Ready
    Status:                True
    Type:                  Ready
  Not After:               2021-07-07T05:45:41Z
  Not Before:              2021-07-02T00:45:41Z
  Renewal Time:            2021-07-04T16:45:41Z
  Revision:                1
Events:                    <none>
```

7. 次のコマンドおよび例を使用して、入力 CRD TLS オプションを編集し、新しい証明書シークレットを指定します。括弧 <> のプレースホルダ値を適切な情報に置き換えます。

```
kubectl edit ingressroutes.traefik.containo.us -n <ACC-deployment-
namespace>
....

# tls:
#    options:
#      name: default
#    secretName: secure-testing-cert
#    store:
#      name: default

 tls:
    options:
      name: default
    secretName: <certificate-secret-name>
    store:
      name: default
```

8. Web ブラウザを使用して、 Astra Control Center の導入 IP アドレスにアクセスします。
9. 証明書の詳細がインストールした証明書の詳細と一致していることを確認します。
10. 証明書をエクスポートし、結果を Web ブラウザの証明書マネージャにインポートします。

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。