はじめに
Astra Control Center

NetApp
August 11, 2025

目次

はじめに

Astra Controlの詳細をご確認ください

Astra Control は、 Kubernetes アプリケーションデータライフサイクル管理解決策で、ステートフルアプリケーションの運用を簡易化します。Kubernetesワークロードの保護、バックアップ、複製、移行を簡易化し、作業アプリケーションのクローンを瞬時に作成できます。

の機能

Astra Control は、 Kubernetes アプリケーションデータのライフサイクル管理に不可欠な機能を提供

- 永続的ストレージを自動的に管理
- アプリケーション対応のオンデマンドの Snapshot とバックアップを作成
- ポリシーベースのスナップショットおよびバックアップ操作を自動化します
- Kubernetes クラスタ間でアプリケーションとデータを移行
- NetApp SnapMirrorテクノロジを使用してアプリケーションをリモートシステムにレプリケート（Astra Control Center）
- ステージング環境から本番環境へのアプリケーションのクローニング
- アプリケーションの稼働状態と保護状態を視覚化します
- Web UIまたはAPIを使用して、バックアップと移行のワークフローを実装します

導入モデル

Astra Control には、次の 2 つの導入モデルがあります。

- * Astra Control Service *：ネットアップが管理するサービス。複数のクラウドプロバイダ環境や自己管理型のKubernetesクラスタで、アプリケーションに対応したデータ管理を提供します。
- * Astra Control Center * ：オンプレミス環境で実行される Kubernetes クラスタのアプリケーション対応データ管理を提供する、自己管理ソフトウェアです。また、NetApp Cloud Volumes ONTAPストレージバックエンドを使用する複数のクラウドプロバイダ環境にAstra Control Centerをインストールすることもできます。

	Astra 制御サービス	**Astra Control Center** の略
どのような方法で提供されますか？	ネットアップのフルマネージドクラウドサービス	ソフトウェアとしてダウンロード、インストール、および管理できます
ホストされているのはどこですか？	ネットアップが選択したパブリッククラウドで実現	自社所有のKubernetesクラスタ
更新方法	管理はネットアップが行います	更新を管理します

	Astra 制御サービス	Astra Control Center の略
サポートされているストレージバックエンドは何ですか。	• Amazon Web Servicesの特長 ◦ Amazon EBSのことです ◦ NetApp ONTAP 対応の Amazon FSX ◦ "Cloud Volumes ONTAP" • Google Cloud ◦ Google Persistent Disk のことです ◦ NetApp Cloud Volumes Service の略 ◦ "Cloud Volumes ONTAP" • Microsoft Azure ◦ Azure Managed Disksの略 ◦ Azure NetApp Files の特長 ◦ "Cloud Volumes ONTAP" • 自己管理クラスタ： ◦ Amazon EBSのことです ◦ Google Persistent Disk のことです ◦ Azure Managed Disksの略 ◦ "Cloud Volumes ONTAP"	• NetApp ONTAP AFF および FAS システム • "Cloud Volumes ONTAP"

Astra Control Service の仕組み

Astra Control Service は、常時稼働し、最新の機能で更新される、ネットアップが管理するクラウドサービスです。複数のコンポーネントを利用して、アプリケーションデータのライフサイクル管理を実現します。

Astra Control Service の概要は次のように機能します。

- Astra Control Service の利用を開始するには、クラウドプロバイダをセットアップし、 Astra アカウントに登録します。
  - GKE クラスタでは、 Astra Control Service はを使用します "NetApp Cloud Volumes Service for Google Cloud" または、永続ボリューム用のストレージバックエンドとして Google Persistent Disk を使用します。
  - AKS クラスタの場合、 Astra Control Service はを使用します "Azure NetApp Files の特長" または、永続ボリューム用のストレージバックエンドとしてAzureで管理されているディスクがあります。
  - Amazon EKSクラスタの場合、Astra Control Serviceはを使用します "Amazon Elastic Block Store" または "NetApp ONTAP 対応の Amazon FSX" 永続ボリュームのストレージバックエンドとして。
- 最初の Kubernetes コンピューティングを Astra Control サービスに追加します。Astra Control Service は、次の処理を実行します。
  - バックアップコピーが格納されるクラウドプロバイダアカウントにオブジェクトストアを作成しま

す。

Azure では、 Astra Control Service によって、 BLOB コンテナ用のリソースグループ、ストレージアカウント、およびキーも作成されます。

- クラスタに新しい admin ロールと Kubernetes サービスアカウントを作成します。
- 新しい admin ロールを使用してインストールします "Astra Trident" をクリックして、 1 つ以上のストレージクラスを作成します。
- ネットアップのクラウドサービスストレージサービスをストレージバックエンドとして使用している場合、Astra Control ServiceはAstra Tridentを使用して、アプリケーション用の永続的ボリュームをプロビジョニングします。Amazon EBSまたはAzureで管理されているディスクをストレージバックエンドとして使用している場合は、プロバイダ固有のCSIドライバをインストールする必要があります。インストール手順については、を参照してください "Amazon Web Servicesをセットアップする" および "Azure で管理されているディスクを使用して Microsoft Azure をセットアップする"。

- この時点で、アプリケーションをクラスタに追加できます。永続ボリュームは、新しいデフォルトのストレージクラスでプロビジョニングされます。
- 次に、 Astra Control Service を使用してこれらのアプリケーションを管理し、スナップショット、バックアップ、クローンの作成を開始します。

Astra Controlの無料プランを使用すると、最大10個のネームスペースをアカウントで管理できます。10以上を管理する場合は、無料プランからプレミアムプランにアップグレードして請求を設定する必要があります。

Astra Control Center の仕組み

Astra Control Center は、お客様のプライベートクラウドでローカルに実行されます。

Astra Control Centerは、ONTAP 9.5以上のストレージバックエンドを備えたAstra Tridentベースのストレージクラスで Kubernetesクラスタをサポートします。

クラウド接続環境では、 Cloud Insights を使用して高度なモニタリングとテレメトリを提供します。Cloud Insights 接続がない場合、 Astra Control Center では、限定的な（ 7 日間の指標）監視と計測データを使用できます。また、オープン指標エンドポイントを介して Kubernetes の標準の監視ツール（ Prometheus や Grafana など）にエクスポートすることもできます。

Astra Control Centerは、AutoSupportとActive IQのデジタルアドバイザ（デジタルアドバイザとも呼ばれます）エコシステムに完全に統合されており、ユーザとNetAppサポートにトラブルシューティングと使用状況の情報を提供します。

90日間の組み込み評価用ライセンスを使用して、Astra Control Centerを試用できます。Astra Control Centerの評価中は、Eメールとコミュニティのオプションでサポートを受けることができます。また、製品内サポートダッシュボードから技術情報アーティクルやドキュメントにアクセスすることもできます。

Astra Control Center をインストールして使用するには、一定の要件を満たす必要があります "要件"。

Astra Control Center の概要は次のように機能します。

- Astra Control Center は、ローカル環境にインストールします。方法の詳細については、こちらをご覧ください "Astra Control Center をインストールします"。
- 次のようなセットアップタスクを実行したとします。
  - ライセンスをセットアップする

- 最初のクラスタを追加します。
- クラスタを追加したときに検出されたストレージバックエンドを追加します。
- アプリケーションバックアップを格納するオブジェクトストアバケットを追加します。

方法の詳細については、こちらをご覧ください "Astra Control Center をセットアップします"。

クラスタにアプリケーションを追加できます。また、管理対象のクラスタにすでにアプリケーションがある場合は、Astra Control Centerを使用してそれらを管理できます。次に、Astra Control Centerを使用して、スナップショット、バックアップ、クローン、およびレプリケーション関係を作成します。

を参照してください。

- "Astra Control Service のマニュアル"
- "Astra Control Center のドキュメント"
- "Astra Trident のドキュメント"
- "Astra Control API を使用"
- "Cloud Insights のドキュメント"
- "ONTAP のドキュメント"

Astra Control Center の要件

運用環境、アプリケーションクラスタ、アプリケーション、ライセンス、 Web ブラウザの準備ができているかどうかを検証します。Astra Control Centerを導入して運用するために、お客様の環境が上記の要件を満たしていることを確認してください。

- サポート対象のホストクラスタKubernetes環境
- [ホストクラスタリソースの要件]
- Astra Trident の要件
- [ストレージバックエンド]
- [イメージレジストリ]
- Astra Control Centerのライセンス
- ONTAP ライセンス
- [ネットワーク要件]
- オンプレミス Kubernetes クラスタへの入力
- サポートされている Web ブラウザ
- [アプリケーションクラスタのその他の要件]

サポート対象のホストクラスタ**Kubernetes**環境

Astra Control Centerは、次のKubernetesホスト環境で検証済みです。

Astra Control Centerをホストするkubernetes環境が、環境の公式ドキュメントに記載されている基本的なリソース要件を満たしていることを確認します。

ホストクラスタ上の**Kubernetes**ディストリビューション	サポートされるバージョン
Azure Stack HCIで実行されるAzure Kubernetes Service	Azure Stack HCI 21H2および22H2（AKS 1.24.xおよび1.25.xを使用）
Google Anthos	1.14～1.16（を参照） Google Anthos Ingressの要件)
Kubernetes（アップストリーム）	1.25~1.27（Kubernetes 1.25以降にはAstra Trident 22.10以降が必要）
Rancher Kubernetes Engine （ RKE ）	RKE 1.3とRancher Manager 2.6 RKE 1.4とRancher Manager 2.7 RKE 2（v1.24.x）とRancher 2.6 RKE 2（v1.25.x）とRancher 2.7
Red Hat OpenShift Container Platform	4.11~4.13

ホストクラスタリソースの要件

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

- * CPU拡張機能*：ホスティング環境のすべてのノードのCPUでAVX拡張機能が有効になっている必要があります。
- ワーカーノード:少なくとも3つのワーカーノードで、それぞれ4つのCPUコアと12GBのRAMを備えています

Astra Trident の要件

お客様の環境のニーズに固有のAstra Tridentの次の要件を満たしていることを確認します。

- * Astra Control Centerで使用する最小バージョン*：Astra Trident 22.10以降のインストールと設定。
- * SnapMirrorレプリケーション*：SnapMirrorベースのアプリケーションレプリケーション用にインストールされているAstra Trident 22.10以降。
- * Kubernetes 1.25以降のサポート*：Kubernetes 1.25以降のクラスタ用にインストールされたAstra Trident 22.10以降（Kubernetes 1.25以降にアップグレードする前にAstra Trident 22.10にアップグレードする必要があります）
- * Astra Tridentを使用したONTAP 構成*：
  - ストレージクラス：クラスタに少なくとも1つのAstra Tridentストレージクラスを設定します。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、そのストレージクラスがデフォルトで指定された唯一のストレージクラスであることを確認します。
  - ストレージドライバとワーカーノード:ポッドがバックエンドストレージと対話できるように、クラスタ内のワーカーノードに適切なストレージドライバが設定されていることを確認します。Astra

Control Center は、 Astra Trident が提供する次の ONTAP ドライバをサポートしています。

- `ontap-nas`
- `ontap-san`
- `ontap-san-economy` （このストレージクラスタイプではアプリケーションレプリケーションは使用できません）
- `ontap-nas-economy` （Snapshot、レプリケーションポリシー、保護ポリシーは、このストレージクラスタイプでは使用できません）。

ストレージバックエンド

十分な容量を備えたサポート対象のバックエンドがあることを確認してください。

- 必要なストレージバックエンド容量：500GB以上の空き容量
- サポートされるバックエンド：Astra Control Centerは次のストレージバックエンドをサポートします。
  - NetApp ONTAP 9.8以降のAFF、FAS、ASAシステム
  - NetApp ONTAP Select 9.8以降
  - NetApp Cloud Volumes ONTAP 9.8以降
  - Longhorn 1.5.0以降
    - VolumeSnapshotClassオブジェクトを手動で作成する必要があります。を参照してください "Longhornドキュメント" 手順については、を参照し
  - NetApp MetroCluster
    - 管理対象のKubernetesクラスタはストレッチ構成に含まれている必要があります。

ONTAP ライセンス

Astra Control Centerを使用するには、必要な機能に応じて、次のONTAP ライセンスがあることを確認します。

- FlexClone
- SnapMirror：オプション。SnapMirrorテクノロジを使用してリモートシステムにレプリケートする場合にのみ必要です。を参照してください "SnapMirrorのライセンス情報"。
- S3ライセンス：オプション。ONTAP S3バケットにのみ必要です

ONTAP システムに必要なライセンスがあるかどうかを確認するには、を参照してください "ONTAPライセンスを管理します。"。

NetApp MetroCluster

NetApp MetroClusterをストレージバックエンドとして使用する場合は、使用するAstra Tridentドライバで、バックエンドオプションとしてSVM管理LIFを指定する必要があります。

MetroCluster LIFを設定するには、Astra Tridentのドキュメントで各ドライバの詳細を参照してください。

- "SAN"
- "NAS"

イメージレジストリ

Astra Control Centerのビルドイメージをプッシュできる既存のプライベートDockerイメージレジストリが必要です。イメージをアップロードするイメージレジストリの URL を指定する必要があります。

Astra Control Centerのライセンス

Astra Control CenterにはAstra Control Centerライセンスが必要です。Astra Control Centerをインストールすると、4、800 CPUユニットの90日間の評価用ライセンスがすでにアクティブ化されています。容量の追加や評価期間の変更が必要な場合や、フルライセンスにアップグレードする場合は、ネットアップから別の評価用ライセンスまたはフルライセンスを取得できます。アプリケーションとデータを保護するにはライセンスが必要です。

Astra Control Centerは無償トライアルにサインアップして試すことができます。登録することでサインアップできます "こちらをご覧ください"。

ライセンスをセットアップするには、を参照してください "90 日間の評価版ライセンスを使用する"。

ライセンスの機能の詳細については、を参照してください "ライセンス"。

ネットワーク要件

Astra Control Centerが適切に通信できるように運用環境を設定します。次のネットワーク設定が必要です。

- * FQDNアドレス*：Astra Control CenterのFQDNアドレスが必要です。
- インターネットへのアクセス：インターネットに外部からアクセスできるかどうかを判断する必要があります。この処理を行わないと、 NetApp Cloud Insights からの監視データや指標データの受信や、へのサポートバンドルの送信など、一部の機能が制限される可能性があります "NetApp Support Site"。
- ポートアクセス：Astra Control Centerをホストする運用環境は、次のTCPポートを使用して通信します。これらのポートがファイアウォールを通過できることを確認し、 Astra ネットワークからの HTTPS 出力トラフィックを許可するようにファイアウォールを設定する必要があります。一部のポートでは、 Astra Control Center をホストする環境と各管理対象クラスタ（該当する場合はメモ）の両方の接続方法が必要です。

> Astra Control Center はデュアルスタック Kubernetes クラスタに導入でき、 Astra Control Center はデュアルスタック操作用に構成されたアプリケーションとストレージバックエンドを管理できます。デュアルスタッククラスタの要件の詳細については、を参照してください "Kubernetes のドキュメント"。

ソース	宛先	ポート	プロトコル	目的
クライアントPC	Astra Control Center の略	443	HTTPS	UI / API アクセス - Astra Control Center をホストしているクラスタと各管理対象クラスタの間で、このポートが双方向に開いていることを確認します

ソース	宛先	ポート	プロトコル	目的
指標利用者	Astra Control Center ワーカーノード	9 -90だ	HTTPS	メトリックデータ通信 - 各管理対象クラスタが、アストラコントロールセンターをホストしているクラスタ上のこのポートにアクセスできることを確認します （双方向通信が必要）
Astra Control Center の略	Hosted Cloud Insights サービスの略 (https://www.netapp.com/cloud-services/cloud-insights/)	443	HTTPS	Cloud Insights 通信
Astra Control Center の略	Amazon S3 ストレージバケットプロバイダ	443	HTTPS	Amazon S3 ストレージ通信
Astra Control Center の略	NetApp AutoSupport (https://support.netapp.com)	443	HTTPS	NetApp AutoSupport 通信

オンプレミス Kubernetes クラスタへの入力

ネットワーク入力アストラコントロールセンターで使用するタイプを選択できます。デフォルトでは、 Astra Control Center は Astra Control Center ゲートウェイ（サービス / traefik ）をクラスタ全体のリソースとして展開します。また、お客様の環境でサービスロードバランサが許可されている場合は、 Astra Control Center でサービスロードバランサの使用もサポートされます。サービスロードバランサを使用する必要があり、設定していない場合は、MetalLBロードバランサを使用して外部IPアドレスを自動的に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。

ロードバランサは、Astra Control CenterワーカーノードのIPアドレスと同じサブネットにあるIPアドレスを使用する必要があります。

詳細については、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"。

Google Anthos Ingressの要件

Google AnthosクラスタでAstra Control Centerをホストする場合、Google AnthosにはMetalLBロードバランサとIstio Ingressサービスがデフォルトで含まれているため、インストール時にAstra Control Centerの一般的な入力機能を簡単に使用できます。を参照してください "Astra Control Center を設定します" を参照してください。

サポートされている Web ブラウザ

Astra Control Center は、最新バージョンの Firefox 、 Safari 、 Chrome をサポートし、解像度は 1280 x 720 以上です。

アプリケーションクラスタのその他の要件

次のAstra Control Center機能を使用する場合は、次の要件に注意してください。

- アプリケーションクラスタの要件： "クラスタ管理の要件"
  - アプリケーション要件の管理： "アプリケーション管理の要件"
  - アプリケーション・レプリケーションの追加要件： "レプリケーションの前提条件"

次のステップ

を表示します "クイックスタート" 概要（ Overview ）：

Astra Control Center のクイックスタート

ここでは、Astra Control Centerの導入に必要な手順の概要を示します。各ステップ内のリンクから、詳細が記載されたページに移動できます。

Kubernetes クラスタの要件を確認

環境が次の要件を満たしていることを確認します。

- Kubernetesクラスタ*
- "ホストクラスタが運用環境の要件を満たしていることを確認します"
- "オンプレミスKubernetesクラスタでロードバランシングを行うための入力を設定する"

ストレージ統合

- "サポートされているAstra Tridentバージョンが環境に含まれていることを確認します"
- "ワーカーノードを準備します"
- "Astra Tridentストレージバックエンドを設定"
- "Astra Tridentストレージクラスを設定する"
- "Astra Tridentボリュームスナップショットコントローラをインストール"
- "ボリュームSnapshotクラスを作成します"
- ONTAP クレデンシャル*
- "ONTAP クレデンシャルを設定する"

Astra Control Centerをダウンロードしてインストールします

次のインストールタスクを実行します。

- "NetApp Support Site のダウンロードページからAstra Control Centerをダウンロードします"
- ネットアップライセンスファイルを入手します。

- Astra Control Centerを評価する場合は、組み込みの評価用ライセンスがすでに付属しています
- "Astra Control Centerをすでに購入している場合は、ライセンスファイルを生成します"

- "Astra Control Center をインストールします"
- "追加のオプション設定手順を実行します"

3 いくつかの初期セットアップ作業を完了します

開始するには、いくつかの基本的なタスクを実行します。

- "ライセンスを追加します"
- "クラスタ管理のための環境を準備します"
- "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"

4 **Astra Control Center** を使用

Astra Control Centerのセットアップが完了したら、Astra Control UIまたはを使用します "Astra Control API の略" アプリの管理と保護を開始するには：

- "アプリの管理":管理するリソースを定義します。
- "アプリを保護します"：保護ポリシーを構成し、アプリケーションのレプリケーション、クローニング、移行を行います。
- "アカウントを管理"：ユーザ、ロール、LDAP、クレデンシャルなど。
- "必要に応じて、Cloud Insights に接続します"：システムの健全性に関する指標を表示します。

を参照してください。

- "Astra Control API を使用"
- "Astra Control Center をアップグレードします"
- "Astra Controlのヘルプ"

インストールの概要

次の Astra Control Center のインストール手順のいずれかを選択して実行します。

- "標準の手順で Astra Control Center をインストールします"
- "（ Red Hat OpenShift を使用する場合） OpenShift OperatorHub を使用して Astra Control Center をインストールします"
- "Cloud Volumes ONTAP ストレージバックエンドに Astra Control Center をインストールします"

環境によっては、Astra Control Centerのインストール後に追加の設定が必要になる場合があります。

- "インストール後にAstra Control Centerを設定します"

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については展開してください

- * 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制限事項"。
- 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。

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。

Astra Control Centerをダウンロードして展開します

1. 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`）から "Astra Control Centerのダウンロードページ"。
2. （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`）をクリックして、バンドルのシグネチャを確認します。

   展開して詳細を表示

   ```
   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、

3. 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一覧表示します。

   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. 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を現在のパスに移動し、次の名前に変更します。 `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。次のように表示されます。 `acc.manifest.bundle.yaml` ファイルと次のディレクトリ：

   ```
   acc/
   kubectl-astra/
   acc.manifest.bundle.yaml
   ```

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=23.07.0-25
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.07.0-25
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/23.07.0-25/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、Astra Control Centerをインストールするクラスタをkubeconfigで指定していることを確認してください。

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。

ステップのために展開

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

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

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/23.07.0-25`）。

```
kubectl create secret docker-registry astra-registry-cred -n
netapp-acc-operator --docker-server=[your_registry_path] --docker
-username=[username] --docker-password=[token]
```

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]
```

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]
```

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 `ASTRA_IMAGE_REGISTRY` 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 `ASTRA_IMAGE_REGISTRY` 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。

サンプル**astra_control_center_operator_deploy.yaml**の展開

```
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: ASTRA_IMAGE_REGISTRY/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: ASTRA_IMAGE_REGISTRY/acc-operator:23.07.25
        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
```

回答例を表示するには展開します。

```
namespace/netapp-acc-operator created
customresourcedefinition.apiextensions.k8s.io/astracontrolcenters.as
tra.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	`23.07.0-25`

<code>astraAddress</code>

設定	ガイダンス（**Guidance**）	を入力します	例
`astraAddress`	を変更します `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"。 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。	文字列	`astra.example.com`

<code>autoSupport</code>

このセクションで選択した内容によって、NetAppのプロアクティブサポートアプリケーション、デジタルアドバイザ、およびデータの送信先が決まります。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
`autoSupport.enrolled`	または `enrolled` または `url` フィールドを選択する必要があります	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。の設定 `true` 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です `false` 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	`false` （デフォルト値）
`autoSupport.url`	または `enrolled` または `url` 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
`imageRegistry.name`	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	`example.registry.com/astra`
`imageRegistry.secret`	に入力した文字列の場合は必須です `imageRegistry.name' requires a secret.` `IMPORTANT: If you are using a registry that does not require authorization, you must delete this` `secret` ラインの内側 `imageRegistry` 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	`astra-registry-cred`

<code>storageClass</code>

設定	ガイダンス（**Guidance**）	を入力します	例
`storageClass`	を変更します `storageClass` からの値 `ontap-gold` 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 `kubectl get sc` 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（`astra-control-center-<version>.manifest`）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。 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。	文字列	`ontap-gold`

<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は保持されます。	文字列	• `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`	次の入力タイプのいずれかを使用します。 **`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。 **`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。 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	文字列	• `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`	デフォルトでは、Astra で高可用性（HA）が使用されます。 `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。 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。	文字列	• `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`	AstraeControlCenterリソース制限のスケー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。 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 `astraResourcesScalar` 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。	文字列	• `Default` （デフォルト値） • `Off`

<code>additionalValues</code>

23.07のインストールで既知の問題が検出されないようにするには、Astra Control CenterのCRに次の値を追加します。

```
additionalValues:
    polaris-keycloak:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
```

- 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
`crds.externalCertManager`	外部証明書マネージャを使用する場合は、変更します `externalCertManager` 終了： `true`。デフォルト `false` 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。 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。	ブール値	`False` （デフォルト値）
`crds.externalTraefik`	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。	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。

サンプルの**astra_control_center.yaml**を展開します。

```
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:
    polaris-keycloak:
      livenessProbe:
        initialDelaySeconds: 180
      readinessProbe:
        initialDelaySeconds: 180
  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）ネームスペース：

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

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]
```

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、環境がを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ることがあります。

サンプル応答のために展開

```
NAME                                          READY   STATUS
RESTARTS     AGE
acc-helm-repo-6cc7696d8f-pmhm8                1/1     Running     0
9h
activity-597fb656dc-5rd4l                     1/1     Running     0
9h
activity-597fb656dc-mqmcw                     1/1     Running     0
9h
api-token-authentication-62f84                1/1     Running     0
9h
api-token-authentication-68nlf                1/1     Running     0
9h
api-token-authentication-ztgrm                1/1     Running     0
9h
asup-669d4ddbc4-fnmwp                         1/1     Running     1
(9h ago)   9h
authentication-78789d7549-lk686               1/1     Running     0
9h
bucketservice-65c7d95496-24x7l                1/1     Running     3
(9h ago)   9h
cert-manager-c9f9fbf9f-k8zq2                  1/1     Running     0
9h
cert-manager-c9f9fbf9f-qjlzm                  1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-b5qll       1/1     Running     0
9h
cert-manager-cainjector-dbbbd8447-p5whs       1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-4722b         1/1     Running     0
9h
cert-manager-webhook-6f97bb7d84-86kv5         1/1     Running     0
9h
certificates-59d9f6f4bd-2j899                 1/1     Running     0
9h
certificates-59d9f6f4bd-9d9k6                 1/1     Running     0
9h
certificates-expiry-check-28011180--1-8lkxz   0/1     Completed   0
9h
cloud-extension-5c9c9958f8-jdhrp              1/1     Running     0
9h
cloud-insights-service-5cdd5f7f-pp8r5         1/1     Running     0
9h
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-mk9l7                 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） `acc-operator` 進捗状況を監視するログ：

```
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` はです `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  23.07.0-25
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

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。ここでは、一部の一般的な入力コントローラタイプの設定手順の例を示します。

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. 入力の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。

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` 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-66dc99cc56-9ldmt   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 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。

OpenShift OperatorHub を使用して Astra Control Center をインストールします

Red Hat OpenShift を使用する場合は、 Red Hat 認定オペレータを使用して Astra Control Center をインストールできます。この手順を使用して、から Astra Control Center をインストールします "Red Hat エコシステムカタログ" または、 Red Hat OpenShift Container Platform を使用します。

この手順を完了したら、インストール手順に戻ってを実行する必要があります "残りのステップ" インストールが成功したかどうかを確認し、ログオンします。

作業を開始する前に

- 環境前提条件を満たしている： "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"。
- 健全なクラスタオペレータとAPIサービス：
  - OpenShiftクラスタから、すべてのクラスタオペレータが正常な状態にあることを確認します。

    ```
    oc get clusteroperators
    ```

  - OpenShiftクラスタから、すべてのAPIサービスが正常な状態であることを確認します。

    ```
    oc get apiservices
    ```

- * FQDN address *：データセンターのAstra Control CenterのFQDNアドレスを取得します。
- * OpenShift Permissions *：説明されているインストール手順を実行するために必要な権限を取得し、Red Hat OpenShift Container Platformにアクセスします。
- **cert manager configure**済み:クラスタにcertマネージャ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。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。
- * Kubernetes入力コントローラ*：クラスター内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するKubernetes入力コントローラがある場合は、Astra Control Centerで使用するようにセットアップする必要があります。
  a. operatorネームスペースを作成します。

```
oc create namespace netapp-acc-operator
```

b. "セットアップを完了" 入力コントローラのタイプ。

手順

- 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]
- [オペレータインストールページを検索します]
- [オペレータをインストールします]
- Astra Control Center をインストールします

Astra Control Centerをダウンロードして展開します

1. 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`）から "Astra Control Centerのダウンロードページ"。
2. （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`）をクリックして、バンドルのシグネチャを確認します。

   展開して詳細を表示

   ```
   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、

3. 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。

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。次のように表示されます。 `acc.manifest.bundle.yaml` ファイルと次のディレクトリ：

   ```
   acc/
   kubectl-astra/
   acc.manifest.bundle.yaml
   ```

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=23.07.0-25
export DIRECTORYNAME=acc
for astraImageFile in $(ls ${DIRECTORYNAME}/images/*.tar) ; do
astraImage=$(podman load --input ${astraImageFile} | sed 's/Loaded
image: //')
astraImageNoPath=$(echo ${astraImage} | sed 's:.*/::')
podman tag ${astraImageNoPath} ${REGISTRY}/netapp/astra/
${PACKAGENAME}/${PACKAGEVERSION}/${astraImageNoPath}
podman push ${REGISTRY}/netapp/astra/${PACKAGENAME}/${
PACKAGEVERSION}/${astraImageNoPath}
done
```

```
<strong>Podman 3</strong>
```

```
export REGISTRY=<MY_FULL_REGISTRY_PATH>
export PACKAGENAME=acc
export PACKAGEVERSION=23.07.0-25
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/23.07.0-25/image:version
```

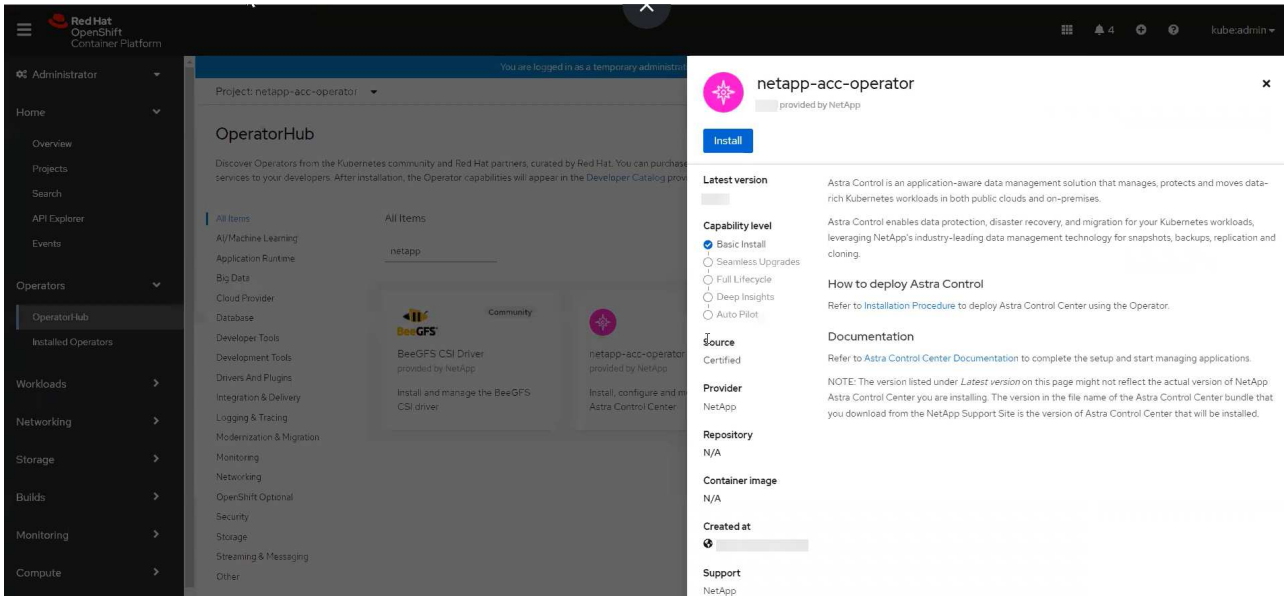
オペレータインストールページを検索します

1. 次のいずれかの手順を実行して、オペレータインストールページにアクセスします。

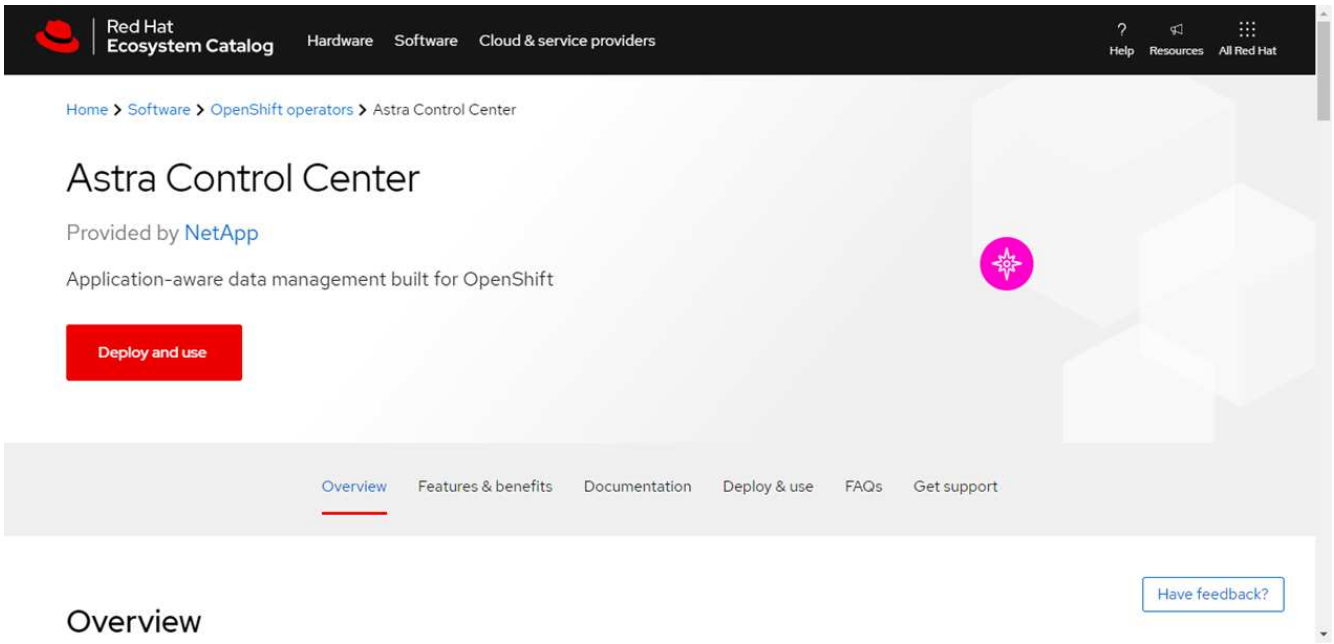
- Red Hat OpenShift の Web コンソールから：
  i. OpenShift Container Platform UI にログインします。
  ii. サイドメニューから、 * 演算子 > OperatorHub * を選択します。

> このオペレータを使用している場合は、Astra Control Centerの最新バージョンにのみアップグレードできます。

  iii. NetApp Astra Control Centerオペレータを検索して選択します。



- Red Hat エコシステムカタログから：
  i. NetApp Astra Control Center を選択します "演算子"。
  ii. [Deploy and Use] を選択します。



オペレータをインストールします

1. 「 * インストールオペレータ * 」ページに必要事項を入力し、オペレータをインストールします。

   オペレータはすべてのクラスタネームスペースで使用できます。

   a. operator名前空間またはを選択します `netapp-acc-operator` オペレータのインストールの一環として、名前空間が自動的に作成されます。

   b. 手動または自動の承認方法を選択します。

      手動による承認が推奨されます。1 つのクラスタで実行する演算子インスタンスは 1 つだけです。

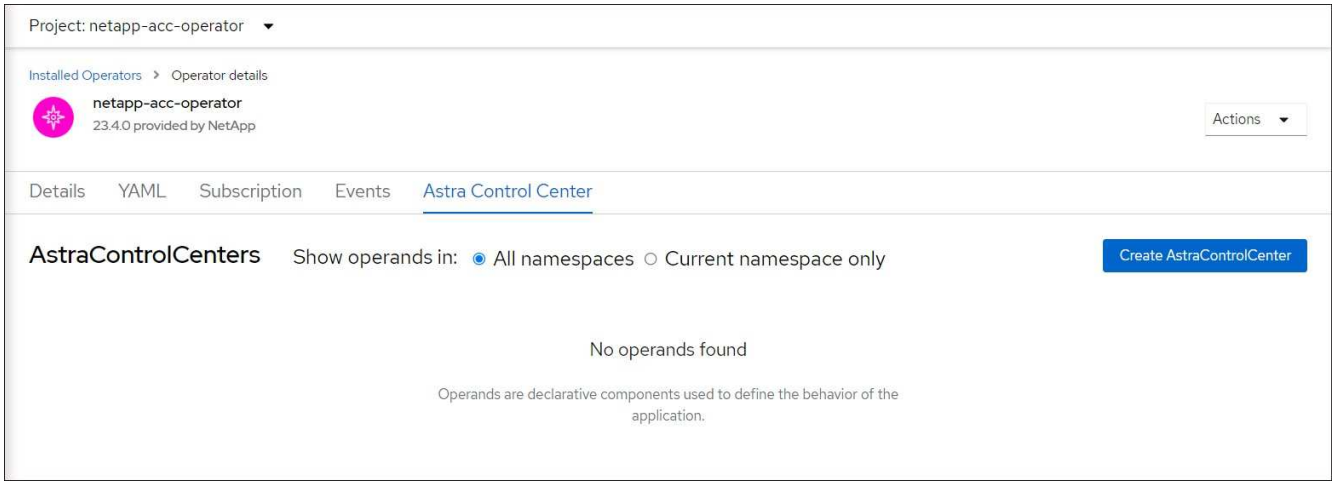
   c. 「 * Install * 」を選択します。

      手動承認方式を選択した場合は、このオペレータの手動インストール計画を承認するように求められます。

2. コンソールで、 OperatorHub メニューに移動して、オペレータが正常にインストールされたことを確認します。

Astra Control Center をインストールします

1. Astra Control Centerオペレータの[**Astra Control Center**]タブ内のコンソールから'[*Create AstraControlCenter *]を選択します



2. を実行します `Create AstraControlCenter` フォームフィールド：

   a. Astra Control Center の名前を保持または調整します。

   b. Astra Control Centerのラベルを追加します。

   c. AutoSupportを有効または無効にします。Auto Support 機能の保持を推奨します。

   d. Astra Control CenterのFQDNまたはIPアドレスを入力します。入らないでください `http://` または `https://` をクリックします。

   e. Astra Control Centerのバージョン（23.07.0-25など）を入力します。

f. アカウント名、 E メールアドレス、および管理者の姓を入力します。

g. ボリューム再利用ポリシーを選択してください `Retain`、 `Recycle`または `Delete`。デフォルト値はです `Retain`。

h. インストールのscaleSizeを選択します。

> デフォルトでは、Astraで高可用性（HA）が使用されます。 `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。

i. 入力タイプを選択します。

- **`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。

- **`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` ゲートウェイをKubernetesの「LoadBalancer」タイプのサービスとして使用します。

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。

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

a. * Image Registry * に、ローカルコンテナイメージのレジストリパスを入力します。入らないでください `http://` または `https://` をクリックします。

b. 認証が必要なイメージレジストリを使用する場合は、イメージシークレットを入力します。

> 認証が必要なレジストリを使用する場合は、 クラスタでシークレットを作成します。

c. 管理者の名を入力します。

d. リソースの拡張を構成する。

e. デフォルトのストレージクラスを指定します。

> 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、そのストレージクラスがデフォルトのアノテーションを持つ唯一のストレージクラスであることを確認します。

f. CRD 処理の環境設定を定義します。

3. YAMLビューを選択して、選択した設定を確認します。
4. 選択するオプション `Create`。

レジストリシークレットを作成します

認証が必要なレジストリを使用する場合は、OpenShiftクラスタでシークレットを作成し、にシークレット名を入力します `Create AstraControlCenter` フォームフィールド。

1. Astra Control Centerオペレータの名前空間を作成します。

```
oc create ns [netapp-acc-operator or custom namespace]
```

2. この名前空間にシークレットを作成します。

```
oc create secret docker-registry astra-registry-cred n [netapp-acc-
operator or custom namespace] --docker-server=[your_registry_path]
--docker username=[username] --docker-password=[token]
```

Astra Controlは、Dockerレジストリシークレットのみをサポートします。

3. の残りのフィールドに値を入力します Create AstraControlCenterフォーム・フィールド。

次のステップ

を実行します "残りのステップ" Astra Control Centerが正常にインストールされたことを確認するには、入力コントローラ（オプション）をセットアップし、UIにログインします。また、を実行する必要があります "セットアップのタスク" インストールが完了したら、

Cloud Volumes ONTAP ストレージバックエンドに Astra Control Center をインストールします

Astra Control Center を使用すると、 Kubernetes クラスタと Cloud Volumes ONTAP インスタンスを自己管理することで、ハイブリッドクラウド環境でアプリケーションを管理できます。Astra Control Center は、オンプレミスの Kubernetes クラスタ、またはクラウド環境内の自己管理型 Kubernetes クラスタのいずれかに導入できます。

これらのいずれかの環境では、 Cloud Volumes ONTAP をストレージバックエンドとして使用して、アプリケーションデータの管理処理を実行できます。バックアップターゲットとして S3 バケットを設定することもできます。

Amazon Web Services（AWS）、Google Cloud Platform（GCP）、および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、クラウド環境に応じて次の手順を実行します。

- Amazon Web Services に Astra Control Center を導入
- Astra Control CenterをGoogle Cloud Platformに導入

- Microsoft Azure に Astra Control Center を導入

OpenShift Container Platform（OCP）などの自己管理型Kubernetesクラスタを使用して、ディストリビューション内のアプリケーションを管理できます。Astra Control Centerを導入するために検証されるのは、自己管理型のOCPクラスタのみです。

Amazon Web Services に Astra Control Center を導入

Amazon Web Services （ AWS ）パブリッククラウドでホストされる自己管理型の Kubernetes クラスタに Astra Control Center を導入できます。

AWSに必要なもの

AWS に Astra Control Center を導入する前に、次のものが必要です。

- Astra Control Center ライセンス。を参照してください "Astra Control Center のライセンス要件"。
- "Astra Control Center の要件を満たす"。
- NetApp Cloud Central アカウント
- OCPを使用する場合は、Red Hat OpenShift Container Platform（OCP）権限（ポッドを作成するためのネームスペースレベル）
- バケットとコネクタを作成するための権限を持つ AWS クレデンシャル、アクセス ID 、シークレットキー
- AWS アカウント Elastic Container Registry （ ECR ）アクセスおよびログイン
- AWS がホストするゾーンと Route 53 エントリは、 Astra Control UI にアクセスするために必要です

AWS の運用環境の要件

Astra Control Center を使用するには、 AWS 向けに次の運用環境が必要です。

- Red Hat OpenShift Container Platform 4.11~4.13

> Astra Control Center をホストするオペレーティングシステムが、環境の公式ドキュメントに記載されている基本的なリソース要件を満たしていることを確認します。

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

コンポーネント	要件
バックエンドの **NetApp Cloud Volumes ONTAP** ストレージ容量	300GB 以上のデータがあります
ワーカーノード（ **AWS EC2** の要件）	少なくとも 3 つのワーカーノードが必要です。 vCPU コア 4 基、 RAM はそれぞれ 12GB です
ロードバランサ	動作環境クラスタ内のサービスに送信される入力トラフィックに使用できるサービスタイプ「 LoadBalancer 」
FQDN	Astra Control Center の FQDN をロードバランシング IP アドレスに指定する方法

コンポーネント	要件
Astra Trident（以前の**Cloud Manager**で、**Kubernetes**クラスタ検出の一部として**NetApp BlueXP**にインストール）	Astra Trident 22.10以降のインストールと設定、およびストレージバックエンドとしてのNetApp ONTAPバージョン9.8以降
イメージレジストリ	NetAppには、Astra Control Centerのビルドイメージの取得に使用できるレジストリが用意されています。 http://netappdownloads.jfrog.io/docker-astra-control-prod Astra Control Centerのインストールプロセスでこのイメージレジストリを使用する手順については、NetAppサポートにお問い合わせください。 NetAppイメージレジストリにアクセスできない場合は、AWS Elastic Container Registry（ECR）などの既存のプライベートレジストリを用意しておく必要があります。このレジストリにAstra Control Centerのビルドイメージをプッシュできます。イメージをアップロードするイメージレジストリの URL を指定する必要があります。 Restic ベースのイメージ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には、 Astra Control Center ホストクラスタと管理対象クラスタが同じイメージレジストリにアクセスできる必要があります。
Astra Trident / ONTAP 構成	Astra Control Center を使用するには、ストレージクラスを作成してデフォルトのストレージクラスとして設定する必要があります。Astra Control Centerは、KubernetesクラスタをNetApp BlueXP（旧Cloud Manager）にインポートするときに作成される次のONTAP Kubernetes ストレージクラスをサポートします。Astra Trident によって提供される機能は次のとおりです。 • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

AWS レジストリトークンは 12 時間で期限切れになり、その後 Docker イメージのレジストリシークレットを更新する必要があります。

AWS の導入の概要を参照してください

Cloud Volumes ONTAP をストレージバックエンドとして使用して Astra Control Center for AWS をインストールするプロセスの概要を以下に示します。

これらの各手順については、以下で詳しく説明します。

1. 十分な IAM 権限があることを確認します。
2. AWS に Red Hat OpenShift クラスタをインストールします。
3. AWSを設定。
4. NetApp BlueXP for AWSを構成します。
5. Astra Control Center for AWSをインストール。

十分な IAM 権限があることを確認します

Red Hat OpenShiftクラスタとNetApp BlueXP（旧Cloud Manager）コネクタをインストールできる十分なIAMロールと権限があることを確認します。

を参照してください "AWS の初期クレデンシャル"。

AWS に Red Hat OpenShift クラスタをインストールします

AWS に Red Hat OpenShift Container Platform クラスタをインストールします。

インストール手順については、を参照してください "AWS で OpenShift Container Platform にクラスタをインストールします"。

AWSを設定

次に、仮想ネットワークを作成するようにAWSを設定し、EC2コンピューティングインスタンスをセットアップし、AWS S3バケットを作成します。にアクセスできない場合 NetApp Astra Control Centerイメージレジストリまた、Astra Control Centerのイメージをホストするelastic Container Registry（ECR）を作成し、このレジストリにイメージをプッシュする必要があります。

AWS のドキュメントに従って次の手順を実行します。を参照してください "AWS インストールドキュメント"。

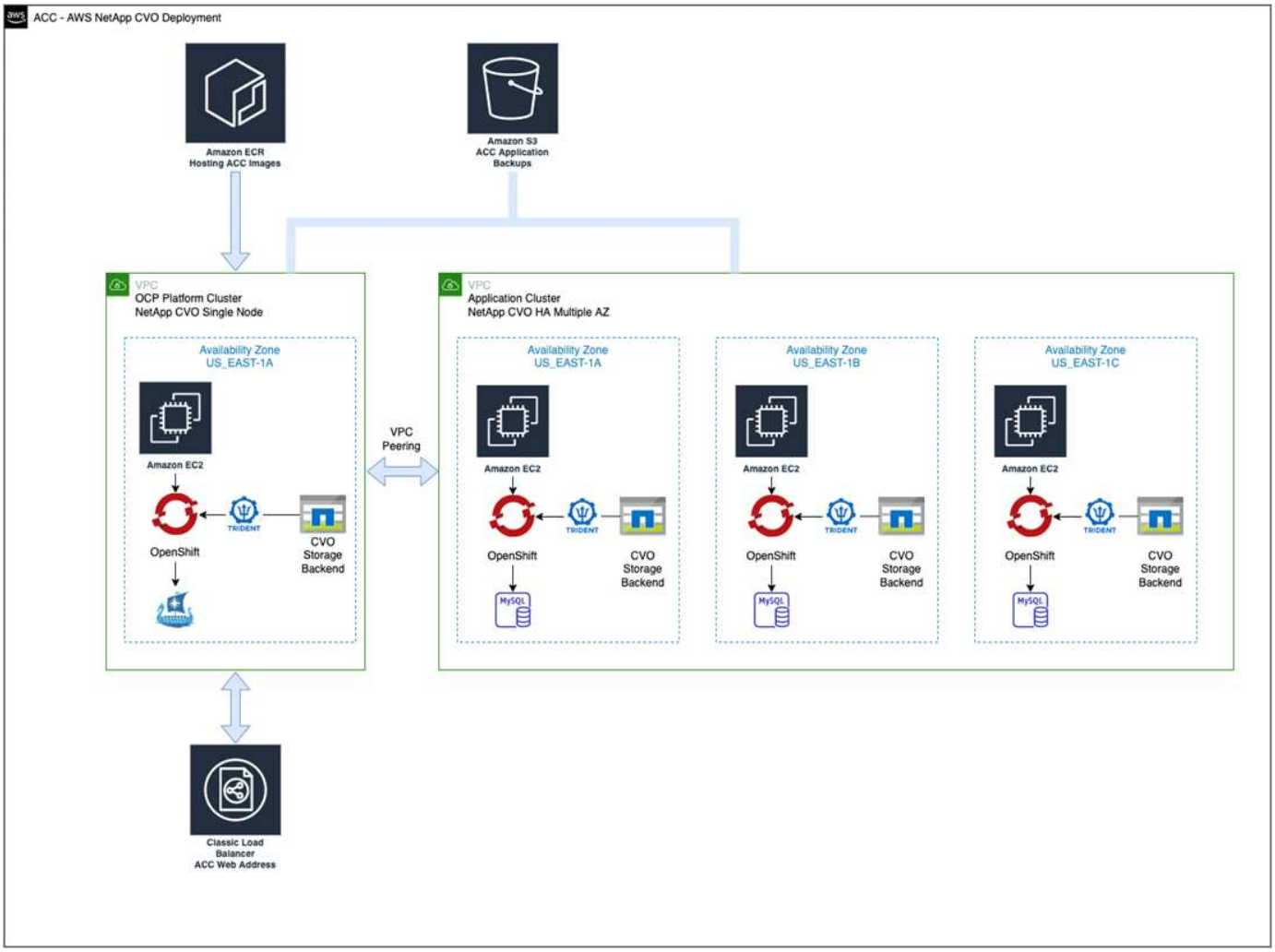
1. AWS仮想ネットワークを作成します。
2. EC2 コンピューティングインスタンスを確認します。AWS ではベアメタルサーバまたは VM を使用できます。
3. インスタンスタイプが、マスターノードとワーカーノードの Astra の最小リソース要件に一致していない場合は、 Astra の要件に合わせて AWS でインスタンスタイプを変更します。 を参照してください "Astra Control Center の要件"。
4. バックアップを格納する AWS S3 バケットを少なくとも 1 つ作成します。
5. （オプション） NetAppイメージレジストリ次の手順を実行します。
    a. AWS Elastic Container Registry（ECR）を作成して、Astra Control Centerのすべてのイメージをホストします。

ECRを作成しないと、Astra Control Centerは、AWSバックエンドを持つCloud Volumes ONTAP を含むクラスタからモニタリングデータにアクセスできません。問題 は、Astra Control Center を使用して検出および管理しようとしたクラスタに AWS ECR アクセスがない場合に発生します。

b. Astra Control Centerのイメージを定義済みのレジストリにプッシュ

AWS Elastic Container Registry （ ECR ）トークンの有効期限は 12 時間です。有効期限が切れたため、クラスタ間のクローニング処理が失敗します。この問題 は、AWS用に設定されたCloud Volumes ONTAP からストレージバックエンドを管理する場合に発生します。この問題 を修正するには、 ECR で再度認証を行い、クローン操作を再開するための新しいシークレットを生成します。

AWS 環境の例を次に示します。



NetApp BlueXP for AWSを構成します

NetApp BlueXP（旧Cloud Manager）を使用して、ワークスペースの作成、AWSへのコネクタの追加、作業環境の作成、クラスタのインポートを行います。

BlueXPのマニュアルに従って'次の手順を実行します以下を参照してください。

- "AWS で Cloud Volumes ONTAP を使用するための準備"。
- "BlueXPを使用してAWSでコネクタを作成します"

手順

1. 資格情報をBlueXPに追加します。
2. ワークスペースを作成します。
3. AWS 用のコネクタを追加します。プロバイダとして AWS を選択します。
4. クラウド環境の作業環境を構築
   a. 場所：「Amazon Web Services（AWS）」
   b. 「 Cloud Volumes ONTAP HA 」と入力します。
5. OpenShift クラスタをインポートします。作成した作業環境にクラスタが接続されます。
   a. ネットアップクラスタの詳細を表示するには、 * K8s * > * Cluster list * > * Cluster Details * を選択します。
   b. 右上にあるAstra Tridentのバージョンを確認します。
   c. Cloud Volumes ONTAP クラスタのストレージクラスは、プロビジョニングツールとしてネットアップを使用していることに注目してください。

      これにより、 Red Hat OpenShift クラスタがインポートされ、デフォルトのストレージクラスに割り当てられます。ストレージクラスを選択します。
      Astra Tridentは、インポートと検出のプロセスで自動的にインストールされます。
6. このCloud Volumes ONTAP 環境内のすべての永続ボリュームとボリュームをメモします。

Cloud Volumes ONTAP は、シングルノードまたはハイアベイラビリティとして動作できます。HA が有効になっている場合は、 AWS で実行されている HA ステータスとノード導入ステータスを確認します。

Astra Control Center for AWSをインストール

標準に従ってください "Astra Control Center のインストール手順"。

AWSでは汎用のS3バケットタイプが使用されます。

Astra Control CenterをGoogle Cloud Platformに導入

Astra Control Centerは、Google Cloud Platform（GCP）パブリッククラウドでホストされる自己管理型のKubernetesクラスタに導入できます。

GCPに必要なもの

GCPでAstra Control Centerを導入する前に、次の項目が必要です。

- Astra Control Center ライセンス。を参照してください "Astra Control Center のライセンス要件"。
- "Astra Control Center の要件を満たす"。
- NetApp Cloud Central アカウント

- OCPを使用している場合、Red Hat OpenShift Container Platform（OCP）4.11~4.13
- OCPを使用する場合は、Red Hat OpenShift Container Platform（OCP）権限（ポッドを作成するためのネームスペースレベル）
- バケットとコネクタの作成を可能にする権限を持つGCPサービスアカウント

GCPの運用環境の要件

Astra Control Center をホストするオペレーティングシステムが、環境の公式ドキュメントに記載されている基本的なリソース要件を満たしていることを確認します。

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

コンポーネント	要件
バックエンドの **NetApp Cloud Volumes ONTAP** ストレージ容量	300GB 以上のデータがあります
ワーカーノード（**GCP**コンピューティング要件）	少なくとも 3 つのワーカーノードが必要です。 vCPU コア 4 基、 RAM はそれぞれ 12GB です
ロードバランサ	動作環境クラスタ内のサービスに送信される入力トラフィックに使用できるサービスタイプ「 LoadBalancer 」
FQDN（**GCP DNS**ゾーン）	Astra Control Center の FQDN をロードバランシング IP アドレスに指定する方法
Astra Trident（以前の**Cloud Manager**で、**Kubernetes**クラスタ検出の一部として**NetApp BlueXP**にインストール）	Astra Trident 22.10以降のインストールと設定、およびストレージバックエンドとしてのNetApp ONTAPバージョン9.8以降[gcp-registry]]
イメージレジストリ	NetAppには、Astra Control Centerのビルドイメージの取得に使用できるレジストリが用意されています。 http://netappdownloads.jfrog.io/docker-astra-control-prod Astra Control Centerのインストールプロセスでこのイメージレジストリを使用する手順については、NetAppサポートにお問い合わせください。 NetAppイメージレジストリにアクセスできない場合は、Google Container Registryなどの既存のプライベートレジストリを用意しておく必要があります。このレジストリにAstra Control Centerのビルドイメージをプッシュできます。イメージをアップロードするイメージレジストリの URL を指定する必要があります。 バックアップ用にリストアイメージを取得するには、匿名アクセスを有効にする必要があります。

コンポーネント	要件
Astra Trident / ONTAP 構成	Astra Control Center を使用するには、ストレージクラスを作成してデフォルトのストレージクラスとして設定する必要があります。Astra Control Centerは、KubernetesクラスタをNetApp BlueXPにインポートするときに作成される次のONTAP Kubernetesストレージクラスをサポートします。Astra Trident によって提供される機能は次のとおりです。 • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

GCPの導入の概要

ここでは、Cloud Volumes ONTAP をストレージバックエンドとして使用して、GCP内の自己管理型OCPクラスタにAstra Control Centerをインストールするプロセスの概要を示します。

これらの各手順については、以下で詳しく説明します。

1. GCPにRed Hat OpenShiftクラスタをインストールします。
2. GCPプロジェクトとVirtual Private Cloudを作成します。
3. 十分な IAM 権限があることを確認します。
4. GCPを設定します。
5. GCP向けNetApp BlueXPの設定。
6. Astra Control Center for GCPをインストールします。

GCPにRed Hat OpenShiftクラスタをインストールします

まず、GCPにRedHat OpenShiftクラスタをインストールします。

インストール手順については、次を参照してください。

- "GCPにOpenShiftクラスタをインストールする"
- "GCPサービスアカウントの作成"

GCPプロジェクトとVirtual Private Cloudを作成します

少なくとも1つのGCPプロジェクトとVirtual Private Cloud（VPC）を作成します。

> OpenShift では、独自のリソースグループを作成できます。さらに、GCP VPCも定義する必要があります。OpenShift のドキュメントを参照してください。

プラットフォームクラスタリソースグループおよびターゲットアプリケーション OpenShift クラスタリソースグループを作成できます。

十分な IAM 権限があることを確認します

Red Hat OpenShiftクラスタとNetApp BlueXP（旧Cloud Manager）コネクタをインストールできる十分なIAMロールと権限があることを確認します。

を参照してください "GCPの初期資格情報と権限"。

GCPを設定します

次に、GCPを設定してVPCを作成し、コンピューティングインスタンスをセットアップし、Google Cloud Object Storageを作成します。にアクセスできない場合 NetApp Astra Control Centerイメージレジストリまた、Astra Control Centerのイメージをホストするgoogle Container Registryを作成し、このレジストリにイメージをプッシュする必要があります。

GCPのドキュメントに従って、次の手順を実行します。「GCPへのOpenShiftクラスタのインストール」を参照してください。

1. GCPでGCPプロジェクトとVPCを作成します。GCPでは、CVOバックエンドでOCPクラスタ用にを使用する予定です。
2. コンピューティングインスタンスを確認します。GCP内のベアメタルサーバまたはVMです。
3. インスタンスタイプが、マスターノードとワーカーノードのAstra最小リソース要件と一致していない場合は、GCPでインスタンスタイプを変更してAstraの要件を満たします。を参照してください "Astra Control Center の要件"。
4. バックアップを保存するGCP Cloud Storageバケットを少なくとも1つ作成します。
5. バケットへのアクセスに必要なシークレットを作成します。
6. （オプション） NetAppイメージレジストリ次の手順を実行します。

    a. Astra Control Centerのイメージをホストするgoogle Container Registryを作成します。

    b. すべてのAstra Control Centerイメージに対して、Dockerプッシュ/プル用のGoogle Container Registryアクセスを設定します。

    例：次のスクリプトを入力して、Astra Control Centerのイメージをこのレジストリにプッシュできます。

    ```
    gcloud auth activate-service-account <service account email address>
    --key-file=<GCP Service Account JSON file>
    ```

    このスクリプトには、Astra Control Centerマニフェストファイルとgoogle Image Registryの場所が必

要です。

例

[+]

```
manifestfile=astra-control-center-<version>.manifest
GCP_CR_REGISTRY=<target image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $GCP_CR_REGISTRY/$image
    docker push $GCP_CR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

7. DNS ゾーンを設定します。

GCP向けNetApp BlueXPの設定

NetApp BlueXP（旧Cloud Manager）を使用して、ワークスペースを作成し、GCPにコネクタを追加し、作業環境を作成して、クラスタをインポートします。

BlueXPのマニュアルに従って'次の手順を実行しますを参照してください "GCPでCloud Volumes ONTAP の使用を開始する"。

作業を開始する前に

- 必要なIAM権限と役割を持つGCPサービスアカウントにアクセスします

手順

1. 資格情報をBlueXPに追加します。を参照してください "GCPアカウントの追加"。
2. GCPのコネクターを追加します。
   a. プロバイダーとして[GCP]を選択します。
   b. GCP資格情報を入力します。を参照してください "BlueXPからGCPでコネクタを作成する"。
   c. コネクタが動作していることを確認し、コネクタに切り替えます。
3. クラウド環境の作業環境を構築
   a. 場所："GCP"
   b. 「 Cloud Volumes ONTAP HA 」と入力します。
4. OpenShift クラスタをインポートします。作成した作業環境にクラスタが接続されます。
   a. ネットアップクラスタの詳細を表示するには、 * K8s * > * Cluster list * > * Cluster Details * を選択します。

b. 右上隅に Trident のバージョンが表示されていることを確認します。

c. Cloud Volumes ONTAP クラスタのストレージクラスは、プロビジョニングツールとして「ネットアップ」を使用していることに注目してください。

これにより、 Red Hat OpenShift クラスタがインポートされ、デフォルトのストレージクラスに割り当てられます。ストレージクラスを選択します。
Astra Tridentは、インポートと検出のプロセスで自動的にインストールされます。

5. このCloud Volumes ONTAP 環境内のすべての永続ボリュームとボリュームをメモします。

Cloud Volumes ONTAP は、シングルノードまたはハイアベイラビリティ（HA）で動作します。HAが有効になっている場合は、GCPで実行されているHAステータスとノード導入ステータスを確認します。

Astra Control Center for GCPをインストールします

標準に従ってください "Astra Control Center のインストール手順"。

GCPでは汎用S3バケットタイプが使用されます。

1. Astra Control Centerインストール用のイメージをプルするDocker Secretを生成します。

```
kubectl create secret docker-registry <secret name> --docker
-server=<Registry location> --docker-username=_json_key --docker
-password="$(cat <GCP Service Account JSON file>)" --namespace=pcloud
```

Microsoft Azure に Astra Control Center を導入

Microsoft Azure パブリッククラウドでホストされる自己管理型の Kubernetes クラスタに Astra Control Center を導入できます。

Azureに必要なもの

Azure に Astra Control Center を導入する前に、次のものが必要です。

- Astra Control Center ライセンス。を参照してください "Astra Control Center のライセンス要件"。
- "Astra Control Center の要件を満たす"。
- NetApp Cloud Central アカウント
- OCPを使用している場合、Red Hat OpenShift Container Platform（OCP）4.11~4.13
- OCPを使用する場合は、Red Hat OpenShift Container Platform（OCP）権限（ポッドを作成するためのネームスペースレベル）
- バケットとコネクタの作成を可能にする権限を持つ Azure クレデンシャル

Azure の運用環境の要件

Astra Control Center をホストするオペレーティングシステムが、環境の公式ドキュメントに記載されている

基本的なリソース要件を満たしていることを確認します。

Astra Control Center では、環境のリソース要件に加え、次のリソースが必要です。

を参照してください "Astra Control Center の運用環境要件"。

コンポーネント	要件
バックエンドの **NetApp Cloud Volumes ONTAP** ストレージ容量	300GB 以上のデータがあります
ワーカーノード（ **Azure** コンピューティング要件）	少なくとも 3 つのワーカーノードが必要です。 vCPU コア 4 基、 RAM はそれぞれ 12GB です
ロードバランサ	動作環境クラスタ内のサービスに送信される入力トラフィックに使用できるサービスタイプ「 LoadBalancer 」
FQDN （ **Azure DNS** ゾーン）	Astra Control Center の FQDN をロードバランシング IP アドレスに指定する方法
Astra Trident（**NetApp BlueXP** の**Kubernetes**クラスタ検出の一部としてインストール）	Astra Trident 22.10以降のインストールと設定、およびNetApp ONTAP バージョン9.8以降がストレージバックエンドとして使用されます
イメージレジストリ	NetAppには、Astra Control Centerのビルドイメージの取得に使用できるレジストリが用意されています。 http://netappdownloads.jfrog.io/docker-astra-control-prod Astra Control Centerのインストールプロセスでこのイメージレジストリを使用する手順については、NetAppサポートにお問い合わせください。 NetAppイメージレジストリにアクセスできない場合は、Azure Container Registry（ACR）などの既存のプライベートレジストリを用意しておく必要があります。このレジストリにAstra Control Centerのビルドイメージをプッシュできます。イメージをアップロードするイメージレジストリの URL を指定する必要があります。 バックアップ用にリストアイメージを取得するには、匿名アクセスを有効にする必要があります。

コンポーネント	要件
Astra Trident / ONTAP 構成	Astra Control Center を使用するには、ストレージクラスを作成してデフォルトのストレージクラスとして設定する必要があります。Astra Control Centerは、KubernetesクラスタをNetApp BlueXPにインポートするときに作成される次のONTAP Kubernetesストレージクラスをサポートします。Astra Trident によって提供される機能は次のとおりです。 • `vsaworkingenvironment-<>-ha-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-ha-san csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-nas csi.trident.netapp.io` • `vsaworkingenvironment-<>-single-san csi.trident.netapp.io`

これらの要件は、運用環境で実行されている唯一のアプリケーションが Astra Control Center であることを前提としています。環境で追加のアプリケーションを実行している場合は、それに応じてこれらの最小要件を調整します。

Azure の導入の概要

ここでは、 Astra Control Center for Azure のインストールプロセスの概要を示します。

これらの各手順については、以下で詳しく説明します。

1. Azure に Red Hat OpenShift クラスタをインストールします。
2. Azure リソースグループを作成する。
3. 十分な IAM 権限があることを確認します。
4. Azure を設定。
5. NetApp BlueXP（旧Cloud Manager）をAzure向けに設定します。
6. Azure向けAstra Control Centerのインストールと設定。

Azure に Red Hat OpenShift クラスタをインストールします

まず、 Azure に Red Hat OpenShift クラスタをインストールします。

インストール手順については、次を参照してください。

- "Azure への OpenShift クラスタのインストール"。
- "Azure アカウントをインストールする"。

Azure リソースグループを作成する

Azure リソースグループを少なくとも 1 つ作成します。

OpenShift では、独自のリソースグループを作成できます。さらに、 Azure リソースグループも定義する必要があります。 OpenShift のドキュメントを参照してください。

プラットフォームクラスタリソースグループおよびターゲットアプリケーション OpenShift クラスタリソースグループを作成できます。

十分な **IAM** 権限があることを確認します

Red Hat OpenShiftクラスタとNetApp BlueXP Connectorをインストールできる十分なIAMロールと権限があることを確認します。

を参照してください "Azure のクレデンシャルと権限"。

Azure を設定

次に、仮想ネットワークを作成し、コンピューティングインスタンスをセットアップし、Azure Blobコンテナを作成するようにAzureを設定します。にアクセスできない場合 NetApp Astra Control Centerイメージレジストリまた、Astra Control Centerのイメージをホストする Azure Container Registry（ACR）を作成し、このレジストリにイメージをプッシュする必要があります。

Azure のドキュメントに従って、次の手順を実行します。を参照してください "Azure への OpenShift クラスタのインストール"。

1. Azure Virtual Networkの作成
2. コンピューティングインスタンスを確認します。Azure の場合、ベアメタルサーバまたは VM を使用できます。
3. インスタンスタイプがまだマスターノードとワーカーノードの Astra 最小リソース要件に一致していない場合は、 Azure でインスタンスタイプを変更して Astra の要件を満たします。 を参照してください "Astra Control Center の要件"。
4. バックアップを格納するAzure BLOBコンテナを少なくとも1つ作成します。
5. ストレージアカウントを作成します。Astra Control Center でバケットとして使用するコンテナを作成するには、ストレージアカウントが必要です。
6. バケットへのアクセスに必要なシークレットを作成します。
7. （オプション） NetAppイメージレジストリ次の手順を実行します。
   a. Astra Control Centerのイメージをホストする Azure Container Registry（ACR）を作成します。
   b. Astra Control Centerのすべてのイメージに対して、Dockerによるプッシュ/プルのACRアクセスをセットアップします。
   c. 次のスクリプトを使用して、Astra Control Centerのイメージをこのレジストリにプッシュします。

```
az acr login -n <AZ ACR URL/Location>
This script requires the Astra Control Center manifest file and your
Azure ACR location.
```

- 例 * ：

```
manifestfile=astra-control-center-<version>.manifest
AZ_ACR_REGISTRY=<target image registry>
ASTRA_REGISTRY=<source Astra Control Center image registry>

while IFS= read -r image; do
    echo "image: $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRY/$image"
    root_image=${image%:*}
    echo $root_image
    docker pull $ASTRA_REGISTRY/$image
    docker tag $ASTRA_REGISTRY/$image $AZ_ACR_REGISTRYY/$image
    docker push $AZ_ACR_REGISTRY/$image
done < astra-control-center-22.04.41.manifest
```

8. DNS ゾーンを設定します。

NetApp BlueXP（旧Cloud Manager）をAzure向けに設定します

BlueXP（旧Cloud Manager）を使用して、ワークスペースの作成、Azureへのコネクタの追加、作業環境の作成、クラスタのインポートを行います。

BlueXPのマニュアルに従って'次の手順を実行しますを参照してください "BlueXPの使用を開始しました"。

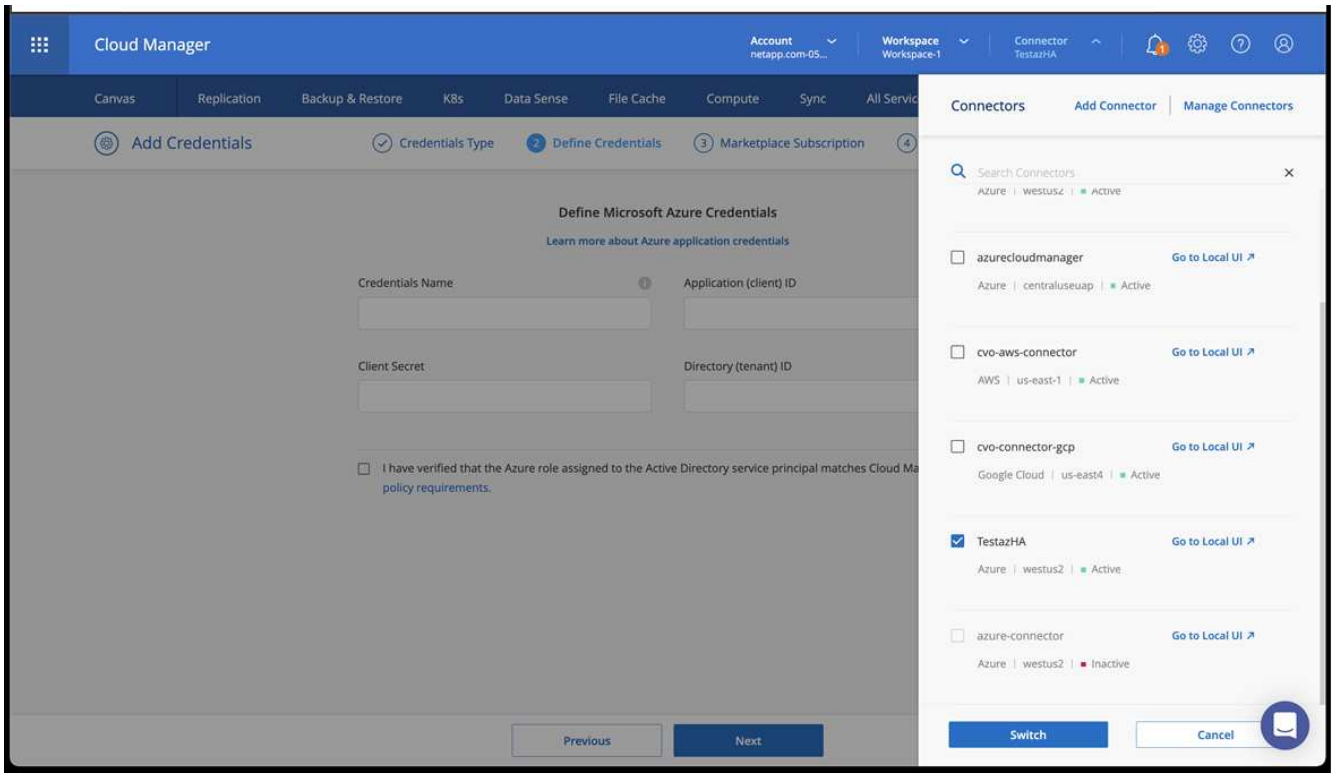
作業を開始する前に

必要な IAM 権限とロールを持つ Azure アカウントにアクセスします

手順

1. 資格情報をBlueXPに追加します。
2. Azure 用のコネクタを追加します。を参照してください "BlueXPポリシー"。
   a. プロバイダとして「 * Azure * 」を選択します。
   b. アプリケーション ID 、クライアントシークレット、ディレクトリ（テナント） ID など、 Azure クレデンシャルを入力します。

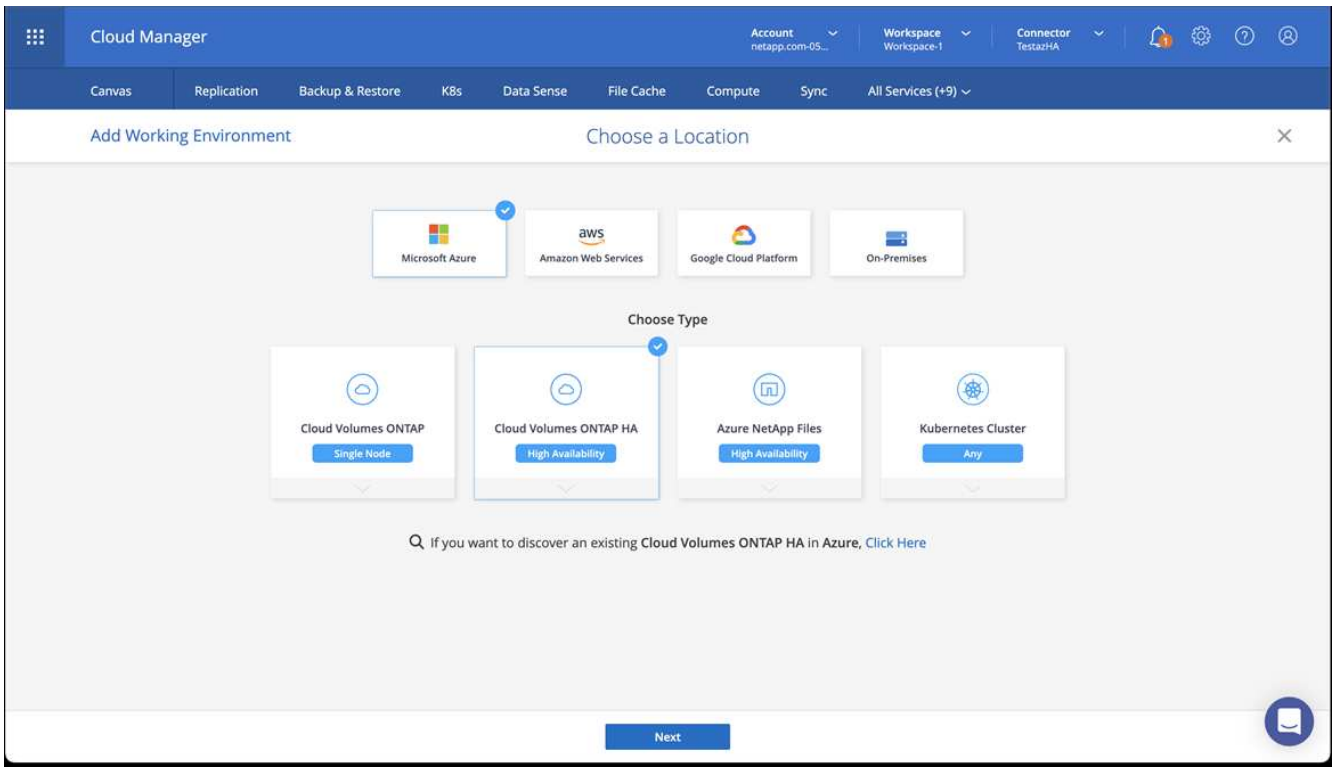
      を参照してください "BlueXPrからAzureでコネクタを作成しています"。
3. コネクタが動作していることを確認し、コネクタに切り替えます。



4. クラウド環境の作業環境を構築

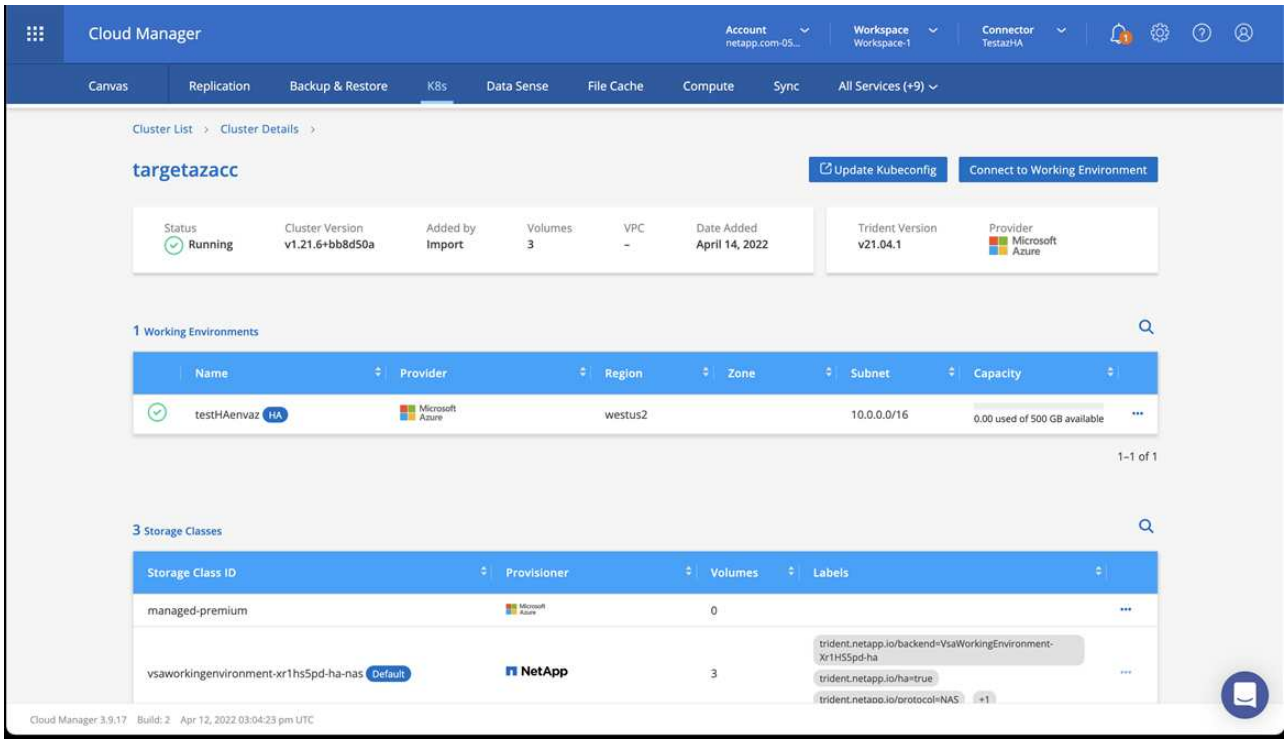
   a. 場所：「 Microsoft Azure 」。

   b. 「 Cloud Volumes ONTAP HA 」と入力します。



5. OpenShift クラスタをインポートします。作成した作業環境にクラスタが接続されます。

a. ネットアップクラスタの詳細を表示するには、 * K8s * > * Cluster list * > * Cluster Details * を選択します。



b. 右上にあるAstra Tridentのバージョンを確認します。

c. Cloud Volumes ONTAP クラスタのストレージクラスは、プロビジョニングツールとしてネットアップを使用していることに注目してください。

これにより、 Red Hat OpenShift クラスタがインポートされ、デフォルトのストレージクラスが割り当てられます。ストレージクラスを選択します。
Astra Tridentは、インポートと検出のプロセスで自動的にインストールされます。

6. このCloud Volumes ONTAP 環境内のすべての永続ボリュームとボリュームをメモします。

7. Cloud Volumes ONTAP は、シングルノードまたはハイアベイラビリティとして動作できます。HA が有効になっている場合は、 Azure で実行されている HA ステータスとノード導入ステータスを確認します。

Azure向けAstra Control Centerのインストールと設定

Astra Control Center を標準でインストールします "インストール手順"。

Astra Control Center を使用して、 Azure バケットを追加する。を参照してください "Astra Control Center をセットアップし、バケットを追加する"。

インストール後に**Astra Control Center**を設定します

環境によっては、Astra Control Centerのインストール後に追加の設定が必要になる場合があります。

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。この方法で環境を構成している場合は、Astra Control Centerをインストールするネームスペースからリソースを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. でリソースクォータを取得します `netapp-acc` （またはカスタム名）ネームスペース：

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

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 or custom
namespace]
```

```
kubectl delete resourcequota pods-low -n [netapp-acc or custom
namespace]
```

```
kubectl delete resourcequota pods-medium -n [netapp-acc or custom
namespace]
```

3. で制限範囲を取得します `netapp-acc` （またはカスタム名）ネームスペース：

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

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 or custom
namespace]
```

カスタム **TLS** 証明書を追加します

Astra Control Centerは、入力コントローラトラフィック（一部の設定のみ）およびWebブラウザでのWeb UI認証に、デフォルトで自己署名TLS証明書を使用します。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。

デフォルトの自己署名証明書は、次の2種類の接続に使用されます。

- Astra Control Center Web UIへのHTTPS接続
- 入力コントローラトラフィック（がの場合のみ） `ingressType: "AccTraefik"` プロパティはで設定されました `astra_control_center.yaml` Astra Control Centerのインストール中にファイルを作成）

これらの接続の認証に使用される証明書は、デフォルトのTLS証明書に置き換えられます。

作業を開始する前に

- Astra Control Center をインストールした Kubernetes クラスタ
- 実行するクラスタ上のコマンドシェルへの管理アクセス `kubectl` コマンド
- 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>` Astra Control Center導入ネームスペースを使用して、次の作業を行います。

```
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>
```

コマンドラインを使用して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、を変更します `spec.selfSigned` の値 `spec.ca.secretName` 以前に作成したTLSシークレットを参照するには、次の手順を実行します

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

CRD：

```
#spec:
#  selfSigned: {}

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

3. 次のコマンドと出力例を使用して、変更が正しいこと、および交換する証明書をクラスタで検証する準備ができていることを確認します `<ACC-deployment-namespace>` Astra Control Center導入ネームスペースを使用して、次の作業を行います。

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

対応：

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

4. を作成します `certificate.yaml` 次の例を使用してファイルを作成し、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:
  <strong>name: <certificate-name></strong>
  namespace: <ACC-deployment-namespace>
spec:
  <strong>secretName: <certificate-secret-name></strong>
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  dnsNames:
  <strong>- <astra.dnsname.example.com></strong> #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>
```

対応：

```
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. 次のコマンドと例を使用してTLS Stores CRDを編集し、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 tlsstores.traefik.io -n <ACC-deployment-namespace>
```

CRD：

```
...
spec:
  defaultCertificate:
    secretName: <certificate-secret-name>
```

8. 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.io -n <ACC-deployment-namespace>
```

CRD：

```
...
 tls:
    secretName: <certificate-secret-name>
```

9. 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。
10. 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。
11. 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。

Astra Control Center をセットアップします

Astra Control Centerをインストールし、UIにログインしてパスワードを変更したら、ライセンスのセットアップ、クラスタの追加、認証の有効化、ストレージの管理、バケットの追加を行うことができます。

タスク

- Astra Control Center のライセンスを追加します
- Astra Controlを使用して、クラスタ管理のための環境を準備する
- [クラスタを追加]
- ONTAP ストレージバックエンドで認証を有効にします
- [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

Astra Control Centerをインストールすると、組み込みの評価用ライセンスがすでにインストールされています。Astra Control Centerを評価する場合は、この手順を省略できます。

新しいライセンスは、Astra Control UIまたはを使用して追加できます "Astra Control API の略"。

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、すべての管理対象Kubernetesクラスタのワーカーノードに割り当てられたCPUリソースを考慮します。ライセンスはvCPUの使用量に基づいています。ライセンスの計算方法の詳細については、を参照してください "ライセンス"。

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。

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"。

作業を開始する前に

- 新しくインストールしたAstra Control Centerインスタンスへのアクセス。
- 管理者ロールの権限。

- A "ネットアップライセンスファイル" （NLF）。

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で、AutoSupport にデータを送信していない場合は、Astra Control Centerに障害が発生した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。

Astra Controlを使用して、クラスタ管理のための環境を準備する

クラスタを追加する前に、次の前提条件を満たし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、クラスタをAstra Control Centerに追加し、クラスタ管理の役割を作成する準備ができていることを確認する必要もあります。

作業を開始する前に

- クラスタ内のワーカーノードで適切なストレージドライバが設定されていることを確認します。これにより、ポッドがバックエンドストレージと通信できるようになります。
- が環境に合っている "運用環境の要件" Astra TridentとAstra Control Centerに最適。
- プライベート認証局（CA）を参照するkubeconfigファイルを使用してクラスタを追加する場合は、 `cluster` kubeconfigファイルのセクションを参照してください。これにより、Astra Controlでクラスタを追加できます。

```
insecure-skip-tls-verify: true
```

- Astra Tridentの一バージョン "Astra Control Centerによってサポートされます" がインストールされている：

可能です "Astra Tridentを導入" Astra Tridentオペレータ（手動またはHelmチャートを使用）またはを使用 `tridentctl`。Astra Tridentのインストールまたはアップグレードを行う前に、を参照してください "サポートされるフロントエンド、バックエンド、およびホスト構成"。

  - * Astra Tridentストレージバックエンドの設定*：Astra Tridentストレージバックエンドが少なくとも1つ必要です "を設定します" クラスタのポリシーを確認してください。
  - * Astra Tridentストレージクラスを設定*：Astra Tridentストレージクラスを少なくとも1つ設定する必要があります "を設定します" クラスタのポリシーを確認してください。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、そのストレージクラスがデフォルトのアノテーションを持つ唯一のストレージクラスであることを確認します。

◦ * Astra Tridentボリュームスナップショットコントローラとボリュームスナップショットクラスがインストールおよび設定されている*：ボリュームスナップショットコントローラがである必要があります "インストール済み" Astra Controlでスナップショットを作成できるようにします。Astra Tridentが少なくとも1つ `VolumeSnapshotClass` はい "セットアップ" 管理者による。

- **Kubeconfig**にアクセス可能:にアクセスできます "デフォルトのクラスタkubeconfig" それは "インストール時に設定"。
- * ONTAP クレデンシャル*：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リストアするには、バックアップONTAP システムでONTAP クレデンシャルとスーパーユーザーIDを設定する必要があります。

  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 sys
export-policy rule modify -vserver <storage virtual machine name>
-policyname <policy name> -ruleindex 1 -anon 65534
```

- **rancher**のみ: Rancher環境でアプリケーションクラスタを管理する場合、rancherから提供されたkubeconfigファイルでアプリケーションクラスタのデフォルトコンテキストを変更して、rancher APIサーバコンテキストではなくコントロールプレーンコンテキストを使用します。これにより、 Rancher APIサーバの負荷が軽減され、パフォーマンスが向上します。

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. Astra Tridentのバージョンを確認

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

Astra 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   23.XX.X
```

Astra 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"
```

Astra 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、続行する前に最新バージョンのAstra 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 "Astra Trident のドキュメント" 手順については、を参照し

2. ポッドが実行されていることを確認します。

```
kubectl get pods -n trident
```

3. サポートされているAstra Tridentドライバをストレージクラスで使用しているかどうかを確認プロビジョニング担当者の名前はとします `csi.trident.netapp.io`。次の例を参照してください。

```
kubectl get sc
```

回答例：

```
NAME                    PROVISIONER             RECLAIMPOLICY
VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
ontap-gold (default)   csi.trident.netapp.io   Delete          Immediate
true                    5d23h
```

クラスタロール**kubeconfig**を作成します。

必要に応じて、Astra Control Center用の限定された権限または拡張された権限管理者ロールを作成できます。kubeconfigはAstra Control Centerのセットアップですでに設定されているため、これは必須の手順ではありません。 "インストールプロセス"。

この手順を使用すると、次のいずれかのシナリオで環境を環境化する場合に、別のkubeconfigを作成できます。

- 管理対象のクラスタに対するAstra Controlの権限を制限する
- 複数のコンテキストを使用し、インストール時に設定されたデフォルトのAstra Control kubeconfigは使用できません。また、単一のコンテキストを持つ限定されたロールは環境では機能しません。

作業を開始する前に

手順 の手順を実行する前に、管理するクラスタに次の情報があることを確認してください。

- kubectl v1.23以降がインストールされている
- Astra Control Centerを使用して追加および管理するクラスタへのアクセス

この手順 では、Astra Control Centerを実行しているクラスタにkubectlでアクセスする必要はありません。

- アクティブなコンテキストのクラスタ管理者の権限で管理するクラスタのアクティブなkubeconfigです

手順

1. サービスアカウントを作成します。

   a. 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 `astracontrol-service-account.yaml`。

      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. 次のいずれかのクラスタロールを作成し、Astra Controlで管理するクラスタに必要な権限を割り当てます。

   ◦ クラスタロールの制限：このロールには、Astra Controlでクラスタを管理するために必要な最小限の権限が含まれます。

ステップのために展開

i. を作成します `ClusterRole` という名前のファイル。例： `astra-admin-account.yaml`。

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>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:

# Get, List, Create, and Update all resources
# Necessary to backup and restore all resources in an app
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - get
  - list
  - create
  - patch

# Delete Resources
# Necessary for in-place restore and AppMirror failover
- apiGroups:
  - ""
  - apps
  - autoscaling
  - batch
  - crd.projectcalico.org
  - extensions
  - networking.k8s.io
  - policy
  - rbac.authorization.k8s.io
  - snapshot.storage.k8s.io
  - trident.netapp.io
  resources:
  - configmaps
  - cronjobs
```

```
  - daemonsets
  - deployments
  - horizontalpodautoscalers
  - ingresses
  - jobs
  - namespaces
  - networkpolicies
  - persistentvolumeclaims
  - poddisruptionbudgets
  - pods
  - podtemplates
  - podsecuritypolicies
  - replicasets
  - replicationcontrollers
  - replicationcontrollers/scale
  - rolebindings
  - roles
  - secrets
  - serviceaccounts
  - services
  - statefulsets
  - tridentmirrorrelationships
  - tridentsnapshotinfos
  - volumesnapshots
  - volumesnapshotcontents
  verbs:
  - delete

# Watch resources
# Necessary to monitor progress
- apiGroups:
  - ""
  resources:
  - pods
  - replicationcontrollers
  - replicationcontrollers/scale
  verbs:
  - watch

# Update resources
- apiGroups:
  - ""
  - build.openshift.io
  - image.openshift.io
  resources:
  - builds/details
```

```
  - replicationcontrollers
  - replicationcontrollers/scale
  - imagestreams/layers
  - imagestreamtags
  - imagetags
  verbs:
  - update

# Use PodSecurityPolicies
- apiGroups:
  - extensions
  - policy
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

ii. （OpenShiftクラスタの場合のみ） `astra-admin-account.yaml` ファイルまたは `# Use PodSecurityPolicies` セクション。

```
# OpenShift security
- apiGroups:
  - security.openshift.io
  resources:
  - securitycontextconstraints
  verbs:
  - use
```

iii. クラスタロールを適用します。

```
kubectl apply -f astra-admin-account.yaml
```

- クラスタロールの拡張：Astra Controlで管理するクラスタの権限の拡張が含まれます。このロールは、複数のコンテキストを使用し、インストール時に設定されたデフォルトのAstra Control kubeconfigを使用できない場合や、単一のコンテキストを持つ限定されたロールが環境で機能しない場合に使用できます。

> ⓘ 次のようになります `ClusterRole` 手順はKubernetesの一般的な例です。ご使用の環境に固有の手順については、ご使用のKubernetesディストリビューションのドキュメントを参照してください。

ステップのために展開

i. を作成します `ClusterRole` という名前のファイル。例： `astra-admin-account.yaml`。

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>astra-admin-account.yaml</strong>
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: astra-admin-account
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

ii. クラスタロールを適用します。

```
kubectl apply -f astra-admin-account.yaml
```

3. サービスアカウントへのクラスタロールバインド用に、クラスタロールを作成します。

a. を作成します `ClusterRoleBinding` という名前のファイルです `astracontrol-clusterrolebinding.yaml`。

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: astra-admin-account
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
```

4. トークンシークレットを作成して適用します。

a. という名前のトークンシークレットファイルを作成します。 `secret-astracontrol-service-account.yaml`。

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

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-astracontrol-service-account
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "astracontrol-service-
account"
type: kubernetes.io/service-account-token
```

b. トークンシークレットを適用します。

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

5. トークンシークレットの名前を `secrets` Array（次の例の最後の行）：

```
kubectl edit sa astracontrol-service-account
```

```
apiVersion: v1
imagePullSecrets:
- name: astracontrol-service-account-dockercfg-48xhx
kind: ServiceAccount
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |

{"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},
"name":"astracontrol-service-account","namespace":"default"}}
  creationTimestamp: "2023-06-14T15:25:45Z"
  name: astracontrol-service-account
  namespace: default
  resourceVersion: "2767069"
  uid: 2ce068c4-810e-4a96-ada3-49cbf9ec3f89
secrets:
- name: astracontrol-service-account-dockercfg-48xhx
<strong>- name: secret-astracontrol-service-account</strong>
```

6. サービスアカウントのシークレットを一覧表示します（置き換えます） `<context>` インストールに適したコンテキストを使用して、次の操作を行います。

```
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-48xhx"},
{ "name": "secret-astracontrol-service-account"}
]
```

内の各要素のインデックス `secrets` アレイは0から始まります。上記の例では、のインデックスです `astracontrol-service-account-dockercfg-48xhx` は0、のインデックスです `secret-astracontrol-service-account` は1です。出力で、サービスアカウントシークレットのインデックス番号をメモします。このインデックス番号は次の手順で必要になります。

7. 次のように kubeconfig を生成します。
   a. を作成します `create-kubeconfig.sh` ファイル。交換してください `TOKEN_INDEX` 次のスクリプトの先頭に正しい値を入力します。

```
<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
```

8. （オプション）クラスタにわかりやすい名前にコバーベキューの名前を変更します。

```
mv kubeconfig-sa 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。

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。

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。

作業を開始する前に

- 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. ダッシュボードまたはクラスタメニューのいずれかから移動します。
   - リソースサマリの*ダッシュボード*で、クラスタペインから*追加*を選択します。
   - 左側のナビゲーション領域で、*クラスタ*を選択し、クラスタページから*クラスタの追加*を選択します。
2. 表示された*クラスタの追加*ウィンドウで、をアップロードします `kubeconfig.yaml` の内容をファイルまたは貼り付けます `kubeconfig.yaml` ファイル。

   。 `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*。

   自分で作成する場合は `kubeconfig` ファイルには、* 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" を参照してください `kubeconfig` ファイル。を使用して、制限されたクラスタロールのkubeconfigを作成した場合 上記のプロセスこの手順では、kubeconfigをアップロードまたは貼り付けてください。
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. 「 * 次へ * 」を選択します。
5. このKubernetesクラスタに使用するデフォルトのストレージクラスを選択し、* Next *を選択します。

   ONTAP ストレージをベースとするAstra 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、すべてが良好な場合は、「*追加」を選択します。

結果

クラスタが「* discovering *」状態になり、「Healthy *」に変わります。これで、Astra Control Centerを使用してクラスタを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。数分経っても問題 が解決しない場合は、クラスタに移動してを実行します `oc get pods -n netapp-monitoring` を開始点として指定します。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。

ONTAP ストレージバックエンドで認証を有効にします

Astra Control Centerには、ONTAP バックエンドの認証に次の2つのモードがあります。

- クレデンシャルベースの認証：必要な権限を持つONTAP ユーザのユーザ名とパスワード。ONTAP のバージョンとの互換性を最大限に高めるには、adminやvsadminなどの事前定義されたセキュリティログイン

ロールを使用する必要があります。

- 証明書ベースの認証：Astra Control Centerは、バックエンドにインストールされている証明書を使用してONTAP クラスタと通信することもできます。クライアント証明書、キー、および信頼されたCA証明書を使用する（推奨）。

後で既存のバックエンドを更新して、あるタイプの認証から別の方法に移行することができます。一度にサポートされる認証方式は1つだけです。

クレデンシャルベースの認証を有効にします

Astra Control Centerには、クラスタを対象としたクレデンシャルが必要です `admin` ONTAP バックエンドと通信するため。事前定義された標準のロール（など）を使用する必要があります `admin`。これにより、Astra Control Centerの今後のリリースで使用する機能APIが公開される可能性がある、将来のONTAP リリースとの前方互換性が確保されます。

> カスタムのセキュリティログインロールはAstra Control Centerで作成して使用できますが、推奨されません。

バックエンド定義の例を次に示します。

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "admin",
  "password": "secret"
}
```

クレデンシャルがプレーンテキストで保存されるのは、バックエンド定義のみです。クレデンシャルの知識が必要なのは、バックエンドの作成または更新だけです。そのため、Kubernetes管理者またはストレージ管理者が実行するのは管理者専用の操作です。

証明書ベースの認証を有効にします

Astra Control Centerでは、証明書を使用して新規および既存のONTAP バックエンドと通信できます。バックエンド定義には、次の情報を入力する必要があります。

- `clientCertificate`:クライアント証明書。
- `clientPrivateKey`:関連付けられた秘密鍵。
- `trustedCACertificate`:信頼されたCA証明書。信頼された CA を使用する場合は、このパラメータを指定する必要があります。信頼された CA が使用されていない場合は無視してかまいません。

次のいずれかのタイプの証明書を使用できます。

- 自己署名証明書
- サードパーティの証明書

自己署名証明書による認証を有効にします

一般的なワークフローは次の手順で構成されます。

手順

1. クライアント証明書とキーを生成します。生成時に、認証に使用するONTAP ユーザに共通名（CN）を設定します。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=<common-name>"
```

2. タイプがのクライアント証明書をインストールします `client-ca` とキーをONTAP 入力します。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

3. ONTAP のセキュリティログインロールが証明書認証方式をサポートしていることを確認します。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

4. 生成した証明書を使用して認証をテストします。ONTAP 管理LIF>と<vserver name> を管理のIPと名前に置き換えてください。LIFのサービスポリシーがに設定されていることを確認する必要があります `default-data-management`。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns=http://www.netapp.com/filer/admin version="1.21" vfiler="<vserver-
name>"><vserver-get></vserver-get></netapp>
```

5. 前の手順で得た値を使用して、Astra Control CenterのUIでストレージバックエンドを追加します。

サードパーティの証明書による認証を有効にします

サードパーティの証明書がある場合は、次の手順で証明書ベースの認証を設定できます。

手順

1. 秘密鍵とCSRを生成します。

```
openssl req -new -newkey rsa:4096 -nodes -sha256 -subj "/" -outform pem
-out ontap_cert_request.csr -keyout ontap_cert_request.key -addext
"subjectAltName = DNS:<ONTAP_CLUSTER_FQDN_NAME>,IP:<ONTAP_MGMT_IP>”
```

2. CSRをWindows CA（サードパーティCA）に渡し、署名済み証明書を問題 します。
3. 署名済み証明書をダウンロードし、「ontap_signed_cert.crt」という名前を付けます。
4. Windows CA（サードパーティCA）からルート証明書をエクスポートします。
5. このファイルに名前を付けます `ca_root.crt`

   これで、次の3つのファイルが作成されました。

   - 秘密鍵： `ontap_signed_request.key` （これは、ONTAP のサーバ証明書に対応するキーです。サーバ証明書のインストール時に必要です）。
   - 署名済み証明書： `ontap_signed_cert.crt` （これは、ONTAP の_server certificate_inとも呼ばれます）。
   - ルート**CA**証明書： `ca_root.crt` （これは、ONTAP の_server-ca certificate_inとも呼ばれます）。
6. これらの証明書をONTAP にインストールします。生成してインストールします `server` および `server-ca` ONTAP の証明書。

sample.yamlの展開

```
# Copy the contents of ca_root.crt and use it here.

security certificate install -type server-ca

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----


You should keep a copy of the CA-signed digital certificate for
future reference.

The installed certificate's CA and serial number for reference:

CA:
serial:

The certificate's generated name for reference:


===

# Copy the contents of ontap_signed_cert.crt and use it here. For
key, use the contents of ontap_cert_request.key file.
security certificate install -type server
Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<certificate details>
-----END CERTIFICATE-----

Please enter Private Key: Press <Enter> when done

-----BEGIN PRIVATE KEY-----
<private key details>
-----END PRIVATE KEY-----

Enter certificates of certification authorities (CA) which form the
certificate chain of the server certificate. This starts with the
issuing CA certificate of the server certificate and can range up to
the root CA certificate.
Do you want to continue entering root and/or intermediate
```

```
certificates {y|n}: n

The provided certificate does not have a common name in the subject
field.
Enter a valid common name to continue installation of the
certificate: <ONTAP_CLUSTER_FQDN_NAME>

You should keep a copy of the private key and the CA-signed digital
certificate for future reference.
The installed certificate's CA and serial number for reference:
CA:
serial:
The certificate's generated name for reference:


==
# Modify the vserver settings to enable SSL for the installed
certificate

ssl modify -vserver <vserver_name> -ca <CA>  -server-enabled true
-serial <serial number>       (security ssl modify)

==
# Verify if the certificate works fine:

openssl s_client -CAfile ca_root.crt -showcerts -servername server
-connect <ONTAP_CLUSTER_FQDN_NAME>:443
CONNECTED(00000005)
depth=1 DC = local, DC = umca, CN = <CA>
verify return:1
depth=0
verify return:1
write W BLOCK
---
Certificate chain
0 s:
   i:/DC=local/DC=umca/<CA>

-----BEGIN CERTIFICATE-----
<Certificate details>
```

7. パスワードを使用しない通信用に同じホストのクライアント証明書を作成します。Astra Control Center は、このプロセスを使用してONTAP と通信します。
8. クライアント証明書を生成してONTAP にインストールします。

sample.yamlの展開

```
# Use /CN=admin or use some other account which has privileges.
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout
ontap_test_client.key -out ontap_test_client.pem -subj "/CN=admin"

Copy the content of ontap_test_client.pem file and use it in the
below command:
security certificate install -type client-ca -vserver <vserver_name>

Please enter Certificate: Press <Enter> when done

-----BEGIN CERTIFICATE-----
<Certificate details>
-----END CERTIFICATE-----

You should keep a copy of the CA-signed digital certificate for
future reference.
The installed certificate’s CA and serial number for reference:

CA:
serial:
The certificate’s generated name for reference:


==

ssl modify -vserver <vserver_name> -client-enabled true
(security ssl modify)

# Setting permissions for certificates
security login create -user-or-group-name admin -application ontapi
-authentication-method cert -role admin -vserver <vserver_name>

security login create -user-or-group-name admin -application http
-authentication-method cert -role admin -vserver <vserver_name>

==

#Verify passwordless communication works fine with the use of only
certificates:

curl --cacert ontap_signed_cert.crt  --key ontap_test_client.key
--cert ontap_test_client.pem
https://<ONTAP_CLUSTER_FQDN_NAME>/api/storage/aggregates
{
```

```
"records": [
{
"uuid": "f84e0a9b-e72f-4431-88c4-4bf5378b41bd",
"name": "<aggr_name>",
"node": {
"uuid": "7835876c-3484-11ed-97bb-d039ea50375c",
"name": "<node_name>",
"_links": {
"self": {
"href": "/api/cluster/nodes/7835876c-3484-11ed-97bb-d039ea50375c"
}
}
},
"_links": {
"self": {
"href": "/api/storage/aggregates/f84e0a9b-e72f-4431-88c4-
4bf5378b41bd"
}
}
}
],
"num_records": 1,
"_links": {
"self": {
"href": "/api/storage/aggregates"
}
}
}%
```


9. Astra Control CenterのUIでストレージバックエンドを追加し、次の値を指定します。
   - クライアント証明書：ontap_test_client.pem
   - 秘密鍵：ontap_test_client.key
   - 信頼された**CA**証明書：ontap_signed_cert.crt

ストレージバックエンドを追加します

既存のONTAP ストレージバックエンドをAstra Control Centerに追加して、そのリソースを管理できます。

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。

クレデンシャルまたは証明書認証情報を設定したら、Astra Control Centerに既存のONTAP ストレージバックエンドを追加してリソースを管理できます。

手順

1. 左側のナビゲーション領域のダッシュボードで、* Backends *を選択します。
2. 「 * 追加」を選択します。
3. [Add storage backend]ページの[Use existing]セクションで、* ONTAP *を選択します。
4. 次のいずれかを選択します。
   - 管理者のクレデンシャルを使用：ONTAP クラスタ管理IPアドレスと管理者のクレデンシャルを入力します。クレデンシャルはクラスタ全体のクレデンシャルである必要があります。

     ここで入力するクレデンシャルのユーザは、を持っている必要があります `ontapi` ONTAP クラスタのONTAP System Managerで有効になっているユーザログインアクセス方法。SnapMirrorレプリケーションを使用する場合は、アクセス方法が指定された「admin」ロールのユーザクレデンシャルを適用します `ontapi` および `http`、ソースとデスティネーションの両方のONTAP クラスタ。を参照してください "ONTAP ドキュメントの「ユーザーアカウントの管理」を参照してください" を参照してください。

   - 証明書を使用：証明書をアップロードします `.pem` ファイル、証明書キー `.key` ファイルを指定し、必要に応じて認証局ファイルを指定します。
5. 「 * 次へ * 」を選択します。
6. バックエンドの詳細を確認し、 * Manage * を選択します。

結果

バックエンドがに表示されます `online` リストに概要情報を表示します。

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

バケットは、Astra Control UIまたはを使用して追加できます "Astra Control API の略"。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。

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、Astra Controlにバケットを作成する必要はありません。アプリケーションのSnapshot機能にはバケットは必要ありません。

作業を開始する前に

- Astra Control Centerで管理しているクラスタから到達できるバケット。
- バケットのクレデンシャル。
- 次のタイプのバケット
  - NetApp ONTAP S3の略
  - NetApp StorageGRID S3 の略
  - Microsoft Azure
  - 汎用 S3

Amazon Web Services（AWS）とGoogle Cloud Platform（GCP）では、汎用のS3バケットタイプを使用します。

Astra Control CenterはAmazon S3を汎用の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をサポートしていると主張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。

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。
2. 「 * 追加」を選択します。
3. 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。

4. 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 はバックアップ先として表示されるため、あとでバックアップを作成する際に選択できます。この名前は、保護ポリシーの設定時にも表示されます。

5. 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。
6. [資格情報の選択*]で、[追加]または[*既存の*を使用]タブのいずれかを選択します。
   - 「*追加」を選択した場合：
     i. 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。
     ii. 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。
   - [既存の使用*]を選択した場合：
     i. バケットで使用する既存のクレデンシャルを選択します。
7. 選択するオプション `Add`。

バケットを追加すると、デフォルトのバケットインジケータで1つのバケットがAstra Controlによってマークされます。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。バケットを追加する際、あとでを選択できます "別のデフォルトバケットを設定する"。

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。

- "ローカルユーザとロールを管理します"
- "アプリの管理を開始します"

- "アプリを保護します"
- "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"
- "デフォルトのストレージクラスを変更する"

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"

Astra Control Center に関するよくある質問

この FAQ は、質問に対する簡単な回答を探している場合に役立ちます。

概要

次のセクションでは、 Astra Control Center を使用しているときに発生する可能性のあるその他の質問に対する回答を示します。詳しい説明については、 astra.feedback@netapp.com までお問い合わせください

Astra Control Center へのアクセス

- Astra Control の URL は何であるか。 *

Astra Control Center は、ローカル認証と各環境に固有の URL を使用します。

URLには、ブラウザで、Astra Control Centerをインストールしたときに、Astra_control_center.yamlカスタムリソース（CR）ファイルのspec.astraatAddressフィールドに設定した完全修飾ドメイン名（FQDN）を入力します。emailは、Astra_control_center.yaml CRのspec.emailフィールドで設定した値です。

ライセンス

評価ライセンスを使用しています。フルライセンスに変更するにはどうすればよいですか**?**

フルライセンスに変更するには、ネットアップからネットアップライセンスファイル（NLF）を入手します。

- 手順 *
  1. 左側のナビゲーションから、 * アカウント * > * ライセンス * を選択します。
  2. ライセンスの概要で、ライセンス情報の右側にある[Options]メニューを選択します。
  3. [置換]*を選択します。
  4. ダウンロードしたライセンスファイルを参照し、 * 追加 * を選択します。

*評価ライセンスを使用しています。アプリを管理できますか？ *

はい。評価ライセンス（デフォルトでインストールされている組み込み評価ライセンスを含む）を使用して、

アプリケーションの管理機能をテストできます。評価用ライセンスとフルライセンスでは、機能や機能に違いはありません。評価用ライセンスは、単純に寿命が短くなります。を参照してください "ライセンス" を参照してください。

Kubernetes クラスタを登録しています

- Astra Control に追加したワーカーノードを Kubernetes クラスタに追加する必要があります。どうすればよいですか？ *

新しいワーカーノードを既存のプールに追加できます。これらは Astra Control によって自動的に検出されます。新しいノードが Astra Control に表示されない場合は、新しいワーカーノードでサポートされているイメージタイプが実行されているかどうかを確認します。を使用して、新しいワーカーノードの健全性を確認することもできます `kubectl get nodes` コマンドを実行します

- クラスタの管理を適切に解除するにはどうすればよいですか *
  1. "Astra Control からアプリケーションの管理を解除"。
  2. "Astra Control からクラスタの管理を解除"。
- Kubernetes クラスタを Astra Control から削除した後、アプリケーションとデータはどうなりますか。 *

Astra Control からクラスタを削除しても、クラスタの構成（アプリケーションと永続的ストレージ）は変更されません。このクラスタで作成されたアプリケーションの Snapshot やバックアップを Astra Control で復元することはできません。Astra Control で作成した永続的ストレージのバックアップは Astra Control に残っていますが、リストアには使用できません。

> 他の方法でクラスタを削除する場合は、必ず事前に Astra Control からクラスタを削除してください。Astra Control で管理している間に別のツールを使用してクラスタを削除した場合、原因で Astra Control アカウントに問題が発生する可能性があります。

管理を解除すると、**NetApp Astra Trident**はクラスタから自動的にアンインストールされますか？
Astra Control Centerでクラスタの管理を解除しても、Astra Tridentはクラスタから自動的にアンインストールされません。Astra Trident をアンインストールするには、が必要です "Astra Trident のドキュメントで次の手順を実行します"。

アプリケーションの管理

- Astra Control はアプリケーションを導入できますか。 *

Astra Control はアプリケーションを導入しない。アプリケーションは Astra Control の外部に導入する必要があります。

- アプリケーションを Astra Control から管理しなくなった後、どうなりますか。 *

既存のバックアップまたは Snapshot がすべて削除されます。アプリケーションとデータは引き続き使用できます。管理対象外のアプリケーション、またはそのアプリケーションに属するバックアップや Snapshot では、データ管理操作を実行できません。

- ネットアップ以外のストレージにあるアプリケーションは Astra Control で管理できますか。 *

いいえネットアップ以外のストレージを使用しているアプリケーションは Astra Control で検出できますが、ネットアップ以外のストレージを使用しているアプリケーションは管理できません。

- Astra Control自体を管理する必要がありますか？*

デフォルトでは、管理可能なアプリケーションとしてAstra Control Centerが表示されませんが、別のAstra Control Centerインスタンスを使用してAstra Control Centerインスタンスをバックアップおよびリストアできます。

不健全なポッドはアプリ管理に影響しますか？

いいえ、ポッドの健全性はアプリ管理には影響しません。

データ管理の操作

- アプリケーションは複数の PVS を使用しています。Astra ControlはこれらのPVSのスナップショットとバックアップを作成しますか*

はい。Astra Controlによるアプリケーションのスナップショット操作には、アプリケーションのPVCにバインドされているすべてのPVSのスナップショットが含まれます。

- Astra Control で取得したスナップショットを、別のインターフェイスやオブジェクトストレージから直接管理できますか。 *

いいえAstra Control で作成したスナップショットとバックアップは、 Astra Control でのみ管理できます。

著作権に関する情報

商標に関する情報