はじめに
Astra Automation

NetApp
August 11, 2025

目次

はじめに

作業を開始する前に

次の手順を確認して、 Astra Control REST API の使用を開始するための準備を簡単に行うことができます。

Astra アカウントのクレデンシャルが必要

Astra のクレデンシャルが必要になるのは、 Astra Web ユーザインターフェイスにサインインして、 API トークンを生成する場合です。Astra Control Center を使用すると、これらの資格情報をローカルで管理できます。Astra Control Service を使用すると、アカウントの資格情報に * Auth0 * サービスからアクセスできます。

Kubernetes の基本概念を理解する

Kubernetes のいくつかの基本概念を理解しておく必要があります。を参照してください "Kubernetes の基本概念" を参照してください。

REST の概念と実装を確認

必ず確認してください "コア REST の実装" REST の概念と、 Astra Control REST API の設計に関する詳細については、を参照してください。

詳細はこちらをご覧ください

に示す追加情報のリソースを確認しておく必要があります "その他のリソース"。

API トークンを取得します

Astra Control REST API を使用するには、 Astra API トークンを取得する必要があります。

概要

API トークンは、 Astra の呼び出し元を識別し、すべての REST API 呼び出しに含める必要があります。

- Astra Webユーザインターフェイスを使用してAPIトークンを生成する必要があります。
- トークンを生成するための手順は、どちらのAstra導入モデルでも同じです。Astraへのアクセスに使用されるURLのみが異なります。
- トークンと関連付けられているアクセス許可とともに伝送されるユーザーIDは、トークンを作成したユーザーによって決定されます。
- トークンは 'Authorization’HTTP 要求ヘッダーに含める必要があります
- トークンは作成後に期限切れになることはありません。
- トークンは、 Astra の Web ユーザインターフェイスから取り消すことができます。

関連情報

- "API トークンを取り消します"

Astra API トークンを作成

Astra API トークンを作成する手順を次に示します。

作業を開始する前に

Astra アカウントのクレデンシャルが必要です。

このタスクについて

このタスクは、 Astra Web インターフェイスで API トークンを生成します。また、 API 呼び出しの際にも必要なアカウント ID を取得する必要があります。

手順

1. 次のアカウントクレデンシャルを使用してAstraにサインインします。
   - Astra Controlサービス： "https://astra.netapp.io"
   - Astra Control Center：インストール時に設定したローカル環境のURLを使用します
2. ページの右上にある図のアイコンをクリックし、 * API access * を選択します。
3. ページで*をクリックし、ポップアップウィンドウで[APIトークンの生成]*をクリックします。
4. アイコンをクリックしてトークン文字列をクリップボードにコピーし、エディタに保存します。
5. 同じページにあるアカウント ID をコピーして保存します。

完了後

curlまたはプログラミング言語を使用してAstra Control REST APIにアクセスする場合は、API Bearerトークンを`HTTP`に含める必要があります `Authorization` 要求ヘッダー。

最初のAPI呼び出し

ワークステーションのCLIで簡単なcurlコマンドを問題 して、Astra Control REST APIの使用を開始し、利用可能かどうかを確認することができます。

作業を開始する前に

curlユーティリティがローカルワークステーションで使用可能になっている必要があります。また、 API トークンと関連付けられているアカウント識別子も必要です。詳細については、を参照してください "API トークンを取得します" 。

カールの例

次のcurlコマンドは、Astraユーザのリストを取得します。図示のとおり、適切な$ACCOUNT_IDと$API_TOKENを指定します。

```
curl --request GET  \
--location "https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/users" \
--include \
--header "Content-Type: application/json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 出力例

```
{
  "items": [
    [
    "David",
    "Anderson",
    "844ec6234-11e0-49ea-8434-a992a6270ec1"
    ],
    [
      "Jane",
      "Cohen",
      "2a3e227c-fda7-4145-a86c-ed9aa0183a6c"
    ]
  ],
  "metadata": {}
}
```

Kubernetes の基本概念

Kubernetes の概念については、 Astra REST API の使用時にいくつか関連するものがあります。

オブジェクト

Kubernetes 環境で管理されるオブジェクトは、クラスタの構成を表す永続的なエンティティです。これらのオブジェクトのことから、クラスタワークロードを含むシステムの状態がわかります。

ネームスペース

ネームスペースは、単一のクラスタ内でリソースを分離する手法を提供します。この組織構造は、作業、ユーザ、およびリソースのタイプを分割する場合に便利です。namespace scope_を 持つオブジェクトはネームスペース内で一意である必要があり、 _cluster scope_を 持つオブジェクトはクラスタ全体で一意である必要があります。

ラベル

ラベルは Kubernetes オブジェクトに関連付けることができます。キーと値のペアを使用して属性を記述します。また、 Kubernetes の中核的な処理には含まれない、組織には役立つ任意の組織をクラスタに適用できます。

ワークフローを使用する準備をします

実際の環境で使用する前に、 Astra ワークフローの構成と形式を理解しておく必要があります。

はじめに

a_workflow_ は、特定の管理タスクまたは目標を達成するために必要な 1 つ以上のステップのシーケンスです。Astra Control ワークフローの各手順は、次のいずれかです。

- REST API 呼び出し（ cURL や JSON の例などの詳細を含む）
- 別の Astra ワークフローの呼び出し
- その他の関連タスク（必要な設計決定の実行など）

ワークフローには、各タスクを実行するために必要な主要な手順とパラメータが含まれています。自動化環境をカスタマイズするための出発点となります。

ワークフローには1つのステップのみを含めることができます。これらのシングルステップワークフローは、複数のステップを含むワークフローとは少し異なります。たとえば、明示的なステップ名は削除されます。アクションまたは操作は、ワークフローのタイトルに基づいて明確にする必要があります。

ワークフローのカテゴリ

導入モデルに応じて、幅広い種類の Astra ワークフローが用意されています。Astra Control Center を使用している場合は、インフラワークフローから始めて、管理ワークフローに進みます。Astra Control Service を使用すると、通常は管理ワークフローに直接移動できます。

ワークフローのcURLのサンプルでは、Astra Control ServiceのURLを使用します。オンプレミスの Astra Control Center を使用している場合は、環境に応じて URL を変更する必要があります。

インフラワークフロー

これらのワークフローは、クレデンシャル、バケット、ストレージバックエンドなどの Astra インフラを処理します。Astra Control Center で必要ですが、ほとんどの場合は Astra Control Service でも使用できます。このワークフローでは、マネージドクラスタの構築と保守に必要なタスクを中心に説明します。

管理ワークフロー

これらのワークフローは、管理対象クラスタのセットアップ後に使用できます。管理ワークフローでは、アプリケーションの保護とサポート処理（アプリケーションのバックアップ、リストア、クローニングなど）に重点が置かれます。

共通の入力パラメータ

以下に記載する入力パラメータは、 REST API 呼び出しを示すために使用するすべての cURL サンプルに共通しています。

これらの入力パラメータは汎用的に必要なため、個々のワークフローでは詳しく説明していません。特定のカールの例に追加の入力パラメータが使用される場合は、「 * その他の入力パラメータ * 」セクションで説明します。

パスパラメータ

すべての REST API 呼び出しで使用されるエンドポイントパスには、次のパラメータが含まれています。も参照してください "URL 形式" を参照してください。

アカウント **ID**

これは、 API 処理を実行する Astra アカウントを識別する UUIDv4 値です。を参照してください "API トークンを取得します" アカウント ID の検索の詳細については、を参照してください。

要求ヘッダー

REST API 呼び出しに応じて、いくつかの要求ヘッダーを含める必要があります。

承認

ワークフロー内のすべての API 呼び出しで、ユーザを識別するための API トークンが必要です。トークンは 'Authorization' 要求ヘッダーに含める必要がありますを参照してください "API トークンを取得します" API トークンの生成の詳細については、を参照してください。

コンテンツタイプ

要求の本文に JSON が含まれている HTTP POST 要求と PUT 要求では、 Astra リソースに基づいてメディアタイプを宣言する必要があります。たとえば ' 管理対象アプリケーションのスナップショットを作成するときに 'Content-Type:application/stra-pappSnap+JSOb' というヘッダーを含めることができます

同意します

アストラリソースに基づいて、応答で想定されるコンテンツの特定のメディアタイプを宣言できます。たとえば ' 管理対象アプリケーションのバックアップを一覧表示するときに 'Accept:application/Astra-pappBackup+JSOb' というヘッダーを含めることができますただし、簡単にするために、ワークフロー内のカールサンプルはすべてのメディアタイプに対応しています。

トークンと識別子の表示

cURL の例で使用される API トークンおよびその他の ID 値は不透明で、認識不能な意味はありません。サンプルの読みやすさを向上させるために、実際のトークンと ID 値は使用されません。代わりに、小さい予約済みキーワードが使用されます。これには次のような利点があります。

- cURL と JSON のサンプルは、より明確でわかりやすくなっています。
- すべてのキーワードは角かっこと大文字で同じ形式を使用するため、挿入または抽出する場所とコンテンツをすばやく識別できます。
- 元のパラメータをコピーして実際の配置で使用することはできないため、値は失われません。

変数はBashシェル環境で使用するためにフォーマットされています。各変数はドル記号で始まり、必要に応じて二重引用符で囲みます。これにより、Bashに認識されるようになります。名前には常に大文字が使用されます。

次に、 curl の例で使用される一般的な予約済みキーワードの一部を示します。このリストはすべてを網羅しているわけではなく、必要に応じてその他のキーワードが使用されていその意味はコンテキストに基づいて明確になる必要があります。

キーワード	を入力します	説明
$ACCOUNT_ID	パス	API 処理を実行するアカウントを識別する UUIDv4 値。

キーワード	を入力します	説明
$API_TOKEN	ヘッダー	発信者を識別および認可するベアラトークン。
$APP_ID	パス	API呼び出しのアプリケーションを識別するUUIDv4値。

Bashでの例の使用

ワークフローカールの例を直接使用する場合は、変数に含まれる変数を環境に適した値に更新する必要があります。以下で説明するように、サンプルを手動で編集するか、Bashシェルに依存して置換を行うことができます。

Bashを使用する利点の1つは、curlコマンドごとに1回ではなく、シェルセッションで変数値を一度だけ設定できることです。

手順

1. Linuxまたは同様のオペレーティングシステムで提供されているBashシェルを開きます。
2. 実行するcurlサンプルに含まれる変数値を設定します。例：

```
$API_TOKEN=SGgpXHeco6M8PLxzIlgbztA4k3_eX4UCa842hOXHBFA=
```

3. ワークフローページからcurlの例をコピーし、シェルターミナルに貼り付けます。
4. ENTER*キーを押して、次のタスクを実行します。
   a. 設定した変数値を置き換えます。
   b. curlコマンドを実行します。

オンラインのAPIリファレンス

Astra APIのリファレンスドキュメントにアクセス

HTTPメソッド、入力パラメータ、応答など、Astra Control REST API呼び出しの詳細にアクセスできます。このリファレンスは、 REST API を使用して自動化アプリケーションを開発する場合に役立ちます。

作業を開始する前に

導入時にAstra Webユーザインターフェイスにサインインするには、クレデンシャルが必要です。リファレンスドキュメントにアクセスするための手順は、Astra ControlサービスとAstra Control Centerで同じです。URLだけが異なります。リファレンスドキュメントにアクセスして表示するためにAPIトークンは必要ありません。

手順

1. 次のアカウントクレデンシャルを使用してAstraにサインインします。
   - Astra Controlサービス： "https://astra.netapp.io"
   - Astra Control Center：インストール時に設定したローカル環境のURLを使用します
2. ページの右上にある図のアイコンをクリックし、 * API access * を選択します。

3. ページの上部で、 * API ドキュメント * の下に表示される URL をクリックします。

結果

Swaggerページが新しいウィンドウまたはタブで表示されます。URLには、サインインしたアカウントのアカウントIDが含まれていることに注意してください。

次の手順

必要に応じて、SwaggerページからAPI呼び出しを実行できます。を参照してください "問題Astra REST API コール" を参照してください。

問題**Astra REST API**コール

Astra Control REST API呼び出しは、APIリファレンスドキュメントページから問題できます。

作業を開始する前に

Astraにサインインし、APIリファレンスページにアクセスする必要があります。を参照してください "Astra APIのリファレンスドキュメントにアクセス" を参照してください。REST APIを使用するにはトークンも必要です。を参照してください "API トークンを取得します" APIトークンの生成の詳細については、を参照してください。

手順

1. APIリファレンスページの上部にある*[Authorize]*をクリックします。
2. APIトークンの値をコピーしてポップアップウィンドウのフィールドに貼り付け、**[Authorize]***をクリックし、[Close]*をクリックします。
3. ページを下にスクロールし、目的のAPI呼び出しを開きます。
4. 右クリック*試してみてください*。
5. 同じAPI呼び出し内で下にスクロールします。必要なパラメータ値を指定し、*[Execute]*をクリックしてコールを問題します。

結果

API呼び出しが実行され、HTTPステータスコードが表示されます。