Get started

Astra Automation

NetApp
July 19, 2022

Table of Contents

Get started

Before you begin

You can quickly prepare to get started with the Astra Control REST API by reviewing the steps below.

Have Astra account credentials

You'll need Astra credentials to sign in to the Astra web user interface and generate an API token. With Astra Control Center, you manage these credentials locally. With Astra Control Service the account credentials are accessed through the **Auth0** service.

Become familiar with basic Kubernetes concepts

You should be familiar with several basic Kubernetes concepts. See Basic Kubernetes concepts for more information.

Review REST concepts and implementation

Make sure to review Core REST implementation for information about REST concepts and the details regarding how the Astra Control REST API is designed.

Get more information

You should be aware of the additional information resources as suggested in Additional resources.

Get an API token

You need to obtain an Astra API token to use the Astra Control REST API.

Introduction

An API token identifies the caller to Astra and must be included with every REST API call.

- You can generate an API token using the Astra web user interface.
- The user identity carried with the token is determined by the user creating the token.
- The token must be included in the `Authorization` HTTP request header.
- A token never expires after it is created.
- You can revoke a token at the Astra web user interface.

Related information

- Revoke an API token

Create an Astra API token

The following steps describe how to create an Astra API token.

Before you begin

You need credentials for an Astra account.

About this task

This task generates an API token at the Astra web interface. You should also retrieve the account ID which is also needed when making an API calls.

Steps

1. Sign in to Astra using your account credentials.

   Access the following site for Astra Control Service: https://astra.netapp.io

2. Click the figure icon at the top right of the page and select **API access**.
3. Click **Generate API token** on the page and in the popup window click **Generate API token**.
4. Click the icon to copy the token string to the clipboard and save it in your editor.
5. Copy and save the account id which is available on the same page.

After you finish

When you access the Astra Control REST API through Curl or a programming language, you must include the API bearer token in the HTTP `Authorization` request header.

Hello world

You can issue a simple Curl command at your workstation's CLI to get started using the Astra Control REST API and confirm its availability.

Before you begin

The Curl utility must be available on your local workstation. You must also have an API token and the associated account identifier. See Get an API token for more information.

Curl example

The following Curl command retrieves a list of Astra users. Provide the appropriate <ACCOUNT_ID> and <API_TOKEN> as indicated.

```
curl --location --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users' --header
'Content-Type: application/json' --header 'Authorization: Bearer
<API_TOKEN>'
```

JSON output example

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

Prepare to use the workflows

You should be familiar with the organization and format of the Astra workflows before using them with a live deployment.

Introduction

A *workflow* is a sequence of one or more steps needed to accomplish a specific administrative task or goal. Each step in an Astra Control workflow is one of the following:

- REST API call (with details such as curl and JSON examples)
- Invocation of another Astra workflow
- Miscellaneous related task (such as making a required design decision)

The workflows include the core steps and parameters needed to accomplish each task. They provide a starting point for customizing your automation environment.

Common input parameters

The input parameters described below are common to all the curl samples used to illustrate a REST API call.

Because these input parameters are universally required, they are not described further in the individual workflows. If additional input parameters are used for a specific curl example, they are described in the section **Additional input parameters**.

Path parameters

The endpoint path used with every REST API call includes the following parameters. Also see URL format for more information.

Account ID

This is the UUIDv4 value identifying the Astra account where the API operation runs. See Get an API token for more information about locating your account ID.

Request headers

There are several request headers that you may need to include depending on the REST API call.

Authorization

All the API calls in the workflows need an API token to identify the user. You must include the token in the `Authorization` request header. See Get an API token for more information about generating an API token.

Content type

With the HTTP POST and PUT requests where JSON is included in the request body, you should declare the media type based on the Astra resource. For example, you can include the header `Content-Type: application/astra-appSnap+json` when creating a snapshot for a managed application.

Accept

You can declare the specific media type of the content you expect in the response based on the Astra resource. For example, you can include the header `Accept: application/astra-appBackup+json` when listing the backups for a managed application. However, for simplicity the curl samples in the workflows accept all media types.

Presentation of tokens and identifiers

The API token and other ID values used with the curl examples are opaque with no discernible meaning. And so to improve the readability of the samples, the actual token and ID values are not used. Rather, smaller reserved keywords are used which has several benefits:

- The curl and JSON samples are clearer and easier to understand.
- Because all the keywords use the same format with brackets and capital letters, you can quickly identify the location and content to insert or extract.
- No value is lost because the original parameters cannot be copied and used with an actual deployment.

Here are some of the common reserved keywords used in the curl examples. This list is not exhaustive and additional keywords are used as needed. Their meaning should be obvious based on the context.

Keyword	Type	Description
<ACCOUNT_ID>	Path	The UUIDv4 value identifying the account where the API operation runs.
<API_TOKEN>	Header	The bearer token identifying and authorizing the caller.
<MANAGED_APP_ID>	Path	The UUIDv4 value identifying the managed application for the API call.

Workflow categories

There are two broad categories of Astra workflows available based on your deployment model. If you are using Astra Control Center, you should start with the infrastructure workflows and then proceed to the management workflows. When using Astra Control Service, you can typically go directly to the management workflows.

The curl samples in the workflows use the URL for the Astra Control Service. You need to change the URL when using the on-premises Astra Control Center as appropriate for your environment.

Infrastructure workflows

These workflows deal with the Astra infrastructure, including credentials, buckets, and storage backends. They are needed with Astra Control Center but in most cases can also be used with Astra Control Service. The workflows focus on the tasks required to establish and maintain an Astra managed cluster.

Management workflows

You can use these workflows after you have a managed cluster. The workflows focus on application protection and support operations such as backing up, restoring, and cloning a managed app.

Basic Kubernetes concepts

There are several Kubernetes concepts that are relevant when using the Astra REST API.

Objects

The objects maintained within a Kubernetes environment are persistent entities representing the configuration of the cluster. These objects collectively describe the state of the system including the cluster workload.

Namespaces

Namespaces provide a technique for isolating resources within a single cluster. This organizational structure is useful when dividing the types of work, users, and resources. Objects with a *namespace scope* need to be unique within the namespace, while those with a *cluster scope* must be unique across the entire cluster.

Labels

Labels can be associated with the Kubernetes objects. They describe attributes using key-value pairs and can enforce an arbitrary organization on the cluster which can be useful to an organization but are outside the core Kubernetes operation.