Using Python

Astra Automation

NetApp
February 12, 2024

Table of Contents

Using Python

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK is an open source package you can use to automate an Astra Control deployment. The package is also a valuable resource for learning about the Astra Control REST API, perhaps as part of creating your own automation platform.

For simplicity, the NetApp Astra Control Python SDK will be referred to as the **SDK** throughout the remainder this page.

Two related software tools

The SDK includes two different though related tools which operate at different levels of abstraction when accessing the Astra Control REST API.

Astra SDK

The Astra SDK provides the core platform functionality. It includes a set of Python classes which abstract the underlying REST API calls. The classes support administrative actions on various Astra Control resources, including apps, backups, snapshots, and clusters.

The Astra SDK is one part of the package and is provided in the single `astraSDK.py` file. You can import this file into your environment and use the classes directly.

The **NetApp Astra Control Python SDK** (or just SDK) is the name of the entire package. The **Astra SDK** refers to the core Python classes in the single file `astraSDK.py`.

Toolkit script

In addition to the Astra SDK file, the `toolkit.py` script is also available. This script operates at a higher level of abstraction by providing access to discrete administrative actions defined internally as Python functions. The script imports the Astra SDK and makes calls to the classes as needed.

How to access

You can access the SDK in the following ways.

Python package

The SDK is available at Python Package Index under the name **actoolkit**. The package is assigned a version number and will continue to be updated as needed. You must use the **PiP** package management utility to install the package into your environment.

Once installed, the astraSDK.py classes can be utilized by placing `import astraSDK` in your scripts. Additionally, `actoolkit` can be invoked directly on your command prompt, and is equivalent to `toolkit.py` (`actoolkit list clusters` is the same as `./toolkit.py list clusters`).

See PyPI: NetApp Astra Control Python SDK for more information.

GitHub source code

The SDK source code is also available at GitHub. The repository includes the following:

- `astraSDK.py` (Astra SDK with Python classes)
- `toolkit.py` (higher level function-based script)
- Detailed installation requirements and instructions
- Installation scripts
- Additional documentation

You can clone the GitHub: Netapp/netapp-astra-toolkits repository to your local environment.

Installation and basic requirements

There are several options and requirements to consider as part of installing the package and preparing to use it.

Summary of the installation options

You can install the SDK in one of the following ways:

- Use the prepared Docker: NetApp/astra-toolkits image, which has all necessary dependencies installed, including `actoolkit`
- Use Pip to install the `actoolkit` package from PyPI into your Python environment
- Clone the GitHub repository and copy/modify the two core Python files so they are accessible to your Python client code

Refer to the PyPI and GitHub pages for more information.

Requirements for the Astra Control environment

Whether directly using the Python classes in the Astra SDK or the functions in the `toolkit.py` script, ultimately you'll be accessing the REST API at an Astra Control deployment. Because of this you'll need an Astra account along with an API token. See Before you begin and the other pages in the **Get started** section of this documentation for more information.

Requirements for the NetApp Astra Control Python SDK

The SDK has several prerequisites related to the local Python environment. For example, you must use Python 3.8 or later. In addition, there are several Python packages that are required. See the GitHub repository page or PyPI package page for more information.

Summary of helpful resources

Here are some the resources you'll need to get started.

- PyPI: NetApp Astra Control Python SDK
- GitHub: Netapp/netapp-astra-toolkits
- Docker: NetApp/astra-toolkits

Native Python

Before you begin

Python is a popular development language for datacenter automation. Before using the

native features of Python together with several common packages, you need to prepare the environment and the required input files.

> In addition to accessing the Astra Control REST API directly using Python, NetApp also provides a toolkit package which abstracts the API and removes some of the complexities. See NetApp Astra Control Python SDK for more information.

Prepare the environment

The basic configuration requirements to run the Python scripts are described below.

Python 3

You need to have the latest version of Python 3 installed.

Additional libraries

The **Requests** and **urllib3** libraries must be installed. You can use pip or another Python management tool as appropriate for your environment.

Network access

The workstation where the scripts run must have network access and be able to reach Astra Control. When using Astra Control Service, you must be connected to the internet and be able to connect to the service at `https://astra.netapp.io`.

Identity information

You need a valid Astra account with the account identifier and API token. See Get an API token for more information.

Create the JSON input files

The Python scripts rely on configuration information contained in JSON input files. Sample files are provided below.

> You need to update the samples as appropriate for your environment.

Identity information

The following file contains the API token and Astra account. You need to pass this file to Python scripts using the `-i` (or `--identity`) CLI parameter.

```
{
  "api_token": "kH4CA_uVIa8q9UuPzhJaAHaGlaR7-no901DkkrVjIXk=",
  "account_id": "5131dfdf-03a4-5218-ad4b-fe84442b9786"
}
```

List the apps

You can use the following script to list the applications for your Astra account.

> See Before you begin for an example of the required JSON input file.

```
1 #!/usr/bin/env python3
2
```

```
##-------------------------------------------------------------------------
------
  3 #
  4 # Usage: python3 list_man_apps.py -i identity_file.json
  5 #
  6 # (C) Copyright 2022 NetApp, Inc.
  7 #
  8 # This sample code is provided AS IS, with no support or warranties of
  9 # any kind, including but not limited for warranties of
merchantability
 10 # or fitness of any kind, expressed or implied. Permission to use,
 11 # reproduce, modify and create derivatives of the sample code is
granted
 12 # solely for the purpose of researching, designing, developing and
 13 # testing a software application product for use with NetApp products,
 14 # provided that the above copyright notice appears in all copies and
 15 # that the software application product is distributed pursuant to
terms
 16 # no less restrictive than those set forth herein.
 17 #
 18
##-------------------------------------------------------------------------
------
 19
 20 import argparse
 21 import json
 22 import requests
 23 import urllib3
 24 import sys
 25
 26 # Global variables
 27 api_token = ""
 28 account_id = ""
 29
 30 def get_managed_apps():
 31     ''' Get and print the list of apps '''
 32
 33     # Global variables
 34     global api_token
 35     global account_id
 36
 37     # Create an HTTP session
 38     sess1 = requests.Session()
 39
 40     # Suppress SSL unsigned certificate warning
```

```
    urllib3.disable_warnings(urllib3.exceptions.
InsecureRequestWarning)

    # Create URL
    url1 = "https://astra.netapp.io/accounts/" + account_id +
"/k8s/v2/apps"

    # Headers and response output
    req_headers  = {}
    resp_headers = {}
    resp_data    = {}

    # Prepare the request headers
    req_headers.clear
    req_headers['Authorization'] = "Bearer " + api_token
    req_headers['Content-Type'] = "application/astra-app+json"
    req_headers['Accept'] = "application/astra-app+json"

    # Make the REST call
    try:
        resp1 = sess1.request('get', url1, headers=req_headers,
allow_redirects=True, verify=False)

    except requests.exceptions.ConnectionError:
        print("Connection failed")
        sys.exit(1)

    # Retrieve the output
    http_code = resp1.status_code
    resp_headers = resp1.headers

    # Print the list of apps
    if resp1.ok:
        resp_data = json.loads(resp1.text)
        items = resp_data['items']
        for i in items:
            print(" ")
            print("Name: " + i['name'])
            print("ID: " + i['id'])
            print("State: " + i['state'])
    else:
        print("Failed with HTTP status code: " + str(http_code))

    print(" ")

    # Close the session
```

```
84      sess1.close()
85
86      return
87
88  def read_id_file(idf):
89      ''' Read the identity file and save values '''
90
91      # Global variables
92      global api_token
93      global account_id
94
95      with open(idf) as f:
96          data = json.load(f)
97
98      api_token = data['api_token']
99      account_id = data['account_id']
100
101     return
102
103 def main(args):
104     ''' Main top level function '''
105
106     # Global variables
107     global api_token
108     global account_id
109
110     # Retrieve name of JSON input file
111     identity_file = args.id_file
112
113     # Get token and account
114     read_id_file(identity_file)
115
116     # Issue REST call
117     get_managed_apps()
118
119     return
120
121 def parseArgs():
122         ''' Parse the CLI input parameters '''
123
124         parser = argparse.ArgumentParser(description='Astra REST API -
    List the apps',
125                                 add_help = True)
126         parser.add_argument("-i", "--identity", action="store", dest
    ="id_file", default=None,
127                                 help='(Req) Name of the identity input
```

```
    file', required=True)
128
129         return parser.parse_args()
130
131 if __name__ == '__main__':
132     ''' Begin here '''
133
134     # Parse input parameters
135     args = parseArgs()
136
137     # Call main function
138     main(args)
```

Copyright information

Trademark information