Uso de Python

Astra Automation 22.04

NetApp
June 28, 2024

Tabla de contenidos

Uso de Python

Kit de desarrollo de software Astra Control Python de NetApp

NetApp Astra Control Python SDK es un paquete de código abierto que puede utilizar para automatizar la puesta en marcha de Astra Control. El paquete es también un recurso valioso para conocer la API REST de Astra Control, quizás como parte de la creación de su propia plataforma de automatización.

Por su simplicidad, el SDK NetApp Astra Control Python se conoce como **SDK** durante el resto de esta página.

Dos herramientas de software relacionadas

El SDK incluye dos herramientas diferentes a través de las relacionadas que funcionan en diferentes niveles de abstracción al acceder a la API REST de Astra Control.

SDK de Astra

Astra SDK proporciona las funciones principales de la plataforma. Incluye un conjunto de clases Python que abstraen las llamadas de la API DE REST subyacente. Estas clases admiten acciones administrativas sobre diversos recursos de Astra Control, como aplicaciones, copias de seguridad, instantáneas y clusters.

El Astra SDK forma parte del paquete y se proporciona en un único `astraSDK.py` archivo. Puede importar este archivo al entorno y utilizar las clases directamente.

* NetApp Astra Control Python SDK* (o solo SDK) es el nombre de todo el paquete. **Astra SDK** se refiere a las clases básicas de Python en un único archivo `astraSDK.py`.

Guión del kit de herramientas

Además del archivo Astra SDK, la `toolkit.py` también está disponible el guión. Este script funciona a un nivel más alto de abstracción al proporcionar acceso a acciones administrativas discretas definidas internamente como funciones Python. La secuencia de comandos importa Astra SDK y realiza llamadas a las clases según sea necesario.

Cómo acceder

Puede acceder al SDK de las siguientes maneras.

Paquete Python

SDK está disponible en "Índice de paquetes Python" con el nombre * netapp-astra-kits*. Al paquete se le asigna un número de versión y se seguirá actualizando según sea necesario. Debe utilizar la utilidad de administración de paquetes **PiP** para instalar el paquete en su entorno.

Consulte "PyPI: Kit de desarrollo de software Astra Control Python de NetApp" si quiere más información.

Código fuente GitHub

El código fuente del SDK también está disponible en GitHub. El repositorio incluye lo siguiente:

- `astraSDK.py` (Astra SDK con clases Python)
- `toolkit.py` (script basado en funciones de mayor nivel)
- Instrucciones y requisitos de instalación detallados
- Scripts de instalación
- Documentación adicional

Puede clonar el "GitHub: NetApp/netapp-astra-kits" repositorio en el entorno local.

Requisitos básicos y de instalación

Hay varias opciones y requisitos que se deben considerar como parte de la instalación del paquete y como parte de la preparación para utilizarlo.

Resumen de las opciones de instalación

Puede instalar el SDK de una de las siguientes maneras:

- Utilice PIP para instalar el paquete de PyPI en su entorno Python
- Clone el repositorio de Git Hub y:
  - Ponga en marcha el paquete como contenedor Docker (que incluye todo lo que necesita)
  - Copie los dos archivos principales de Python para que puedan acceder a su código de cliente Python

Consulte las páginas PyPI y GitHub para obtener más información.

Requisitos para el entorno de Astra Control

Ya sea utilizando directamente las clases Python en Astra SDK o las funciones de `toolkit.py` Script, en última instancia, accederá a la API DE REST en una implementación de Astra Control. Gracias a esto, necesitará una cuenta Astra junto con un token de API. Consulte "Antes de empezar" Y las otras páginas de la sección **Introducción** de esta documentación para obtener más información.

Requisitos del SDK de Astra Control Python de NetApp

El SDK tiene varios requisitos previos relacionados con el entorno local de Python. Por ejemplo, debe utilizar Python 3.5 o posterior. Además, hay varios paquetes Python que son necesarios. Consulte la página del repositorio de GitHub o la página del paquete PyPI para obtener más información.

Resumen de recursos útiles

Estos son algunos de los recursos que necesitará para comenzar.

- "PyPI: Kit de desarrollo de software Astra Control Python de NetApp"
- "GitHub: NetApp/netapp-astra-kits"

Python nativo

Antes de empezar

Python es un lenguaje de desarrollo popular especialmente para la automatización del centro de datos. Antes de utilizar las características nativas de Python junto con varios paquetes comunes, debe preparar el entorno y los archivos de entrada necesarios.

Además de acceder a la API DE REST de Astra Control directamente con Python, NetApp también ofrece un paquete de herramientas que abstrae la API y elimina algunas de las complejidades. Consulte "Kit de desarrollo de software Astra Control Python de NetApp" si quiere más información.

Preparar el entorno de

A continuación se describen los requisitos básicos de configuración para ejecutar los scripts de Python.

Python 3

Necesita tener instalada la última versión de Python 3.

Bibliotecas adicionales

Las bibliotecas **Requests** y **urllib3** deben estar instaladas. Puede utilizar pip u otra herramienta de gestión Python según sea necesario para su entorno.

Acceso a la red

La estación de trabajo donde se ejecuten las secuencias de comandos debe tener acceso a la red y poder llegar a Astra Control. Cuando utilice Astra Control Service, debe estar conectado a Internet y poder conectarse al servicio en `https://astra.netapp.io`.

Información de identidad

Necesita una cuenta Astra válida con el identificador de cuenta y el token de API. Consulte "Obtenga un token de API" si quiere más información.

Cree los archivos de entrada JSON

Los scripts Python se basan en la información de configuración contenida en los archivos de entrada JSON. A continuación se proporcionan archivos de ejemplo.

Debe actualizar las muestras según sea necesario para su entorno.

Información de identidad

El siguiente archivo contiene el token de la API y la cuenta de Astra. Debe pasar este archivo a los scripts de Python mediante `-i` (o. `--identity`) Parámetro CLI.

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

Enumere las aplicaciones gestionadas

Puede utilizar la siguiente secuencia de comandos para enumerar las aplicaciones gestionadas de su cuenta Astra.

Consulte "Antes de empezar" Por ejemplo del archivo de entrada JSON requerido.

```
#!/usr/bin/env python3
##-------------------------------------------------------------------------
------
#
# Usage: python3 list_man_apps.py -i identity_file.json
#
# (C) Copyright 2021 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##-------------------------------------------------------------------------
------

import argparse
import json
import requests
import urllib3
import sys

# Global variables
api_token = ""
account_id = ""

def get_managed_apps():
    ''' Get and print the list of managed apps '''

    # Global variables
    global api_token
    global account_id

    # Create an HTTP session
    sess1 = requests.Session()

    # Suppress SSL unsigned certificate warning
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

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

```
"/k8s/v1/managedApps"

    # 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-managedApp+json"
    req_headers['Accept'] = "application/astra-managedApp+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 managed 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
    sess1.close()

    return

def read_id_file(idf):
    ''' Read the identity file and save values '''
```

```
    # Global variables
    global api_token
    global account_id

    with open(idf) as f:
        data = json.load(f)

    api_token = data['api_token']
    account_id = data['account_id']

    return

def main(args):
    ''' Main top level function '''

    # Global variables
    global api_token
    global account_id

    # Retrieve name of JSON input file
    identity_file = args.id_file

    # Get token and account
    read_id_file(identity_file)

    # Issue REST call
    get_managed_apps()

    return

def parseArgs():
        ''' Parse the CLI input parameters '''

        parser = argparse.ArgumentParser(description='Astra REST API -
List the managed apps',
                            add_help = True)
        parser.add_argument("-i", "--identity", action="store", dest
="id_file", default=None,
                            help='(Req) Name of the identity input file',
required=True)

        return parser.parse_args()

if __name__ == '__main__':
    ''' Begin here '''
```

```
    # Parse input parameters
    args = parseArgs()

    # Call main function
    main(args)
```

Información de copyright

Información de la marca comercial