# Python 사용

## Astra Automation

NetApp
August 11, 2025

This PDF was generated from https://docs.netapp.com/ko-kr/astra-automation-2310/python/astra_toolkits.html on August 11, 2025. Always check docs.netapp.com for the latest.

# 목차


Python 사용 . . . 1
NetApp Astra Control Python SDK . . . 1
두 가지 관련 소프트웨어 도구 . . . 1
액세스 방법 . . . 1
설치 및 기본 요구 사항 . . . 2
유용한 리소스 요약 . . . 2
기본 Python . . . 2
시작하기 전에 . . . 3
앱을 나열합니다 . . . 4

# Python 사용

## NetApp Astra Control Python SDK

NetApp Astra Control Python SDK는 Astra Control 구축을 자동화하는 데 사용할 수 있는 오픈 소스 패키지입니다. 또한 이 패키지는 Astra Control REST API에 대해 학습하는 귀중한 리소스로서, 자신의 자동화 플랫폼을 구축하는 데 사용될 수 있습니다.

간소화를 위해 NetApp Astra Control Python SDK를 나머지 페이지 전체에서 * SDK * 라고 부릅니다.

### 두 가지 관련 소프트웨어 도구

SDK에는 Astra Control REST API에 액세스할 때 서로 다른 추상화 수준에서 작동하는 두 가지 서로 다른 관련 툴이 포함되어 있습니다.

#### Astra SDK

Astra SDK는 핵심 플랫폼 기능을 제공합니다. 기본 REST API 호출을 추상화하는 Python 클래스 집합이 포함되어 있습니다. 이 클래스는 앱, 백업, 스냅샷 및 클러스터를 포함한 다양한 Astra Control 리소스에 대한 관리 작업을 지원합니다.

Astra SDK는 패키지의 한 부분으로, 단일 패키지로 제공됩니다 `astraSDK.py` 파일. 이 파일을 사용자 환경으로 가져오고 클래스를 직접 사용할 수 있습니다.

NetApp Astra Control Python SDK * (또는 SDK)는 전체 패키지의 이름입니다. Astra SDK * 는 단일 파일의 핵심 Python 클래스를 나타냅니다 `astraSDK.py`.

툴킷 스크립트입니다

Astra SDK 파일 외에도, `toolkit.py` 스크립트를 사용할 수도 있습니다. 이 스크립트는 Python 함수로 내부적으로 정의된 개별 관리 작업에 대한 액세스를 제공하여 높은 수준의 추상화로 작동합니다. 이 스크립트는 Astra SDK를 가져오고 필요에 따라 클래스를 호출합니다.

### 액세스 방법

다음과 같은 방법으로 SDK에 액세스할 수 있습니다.

#### Python 패키지

SDK는 에서 확인할 수 있습니다 "Python 패키지 색인" 라는 이름으로 * actoolkit * 를 사용합니다. 패키지에 버전 번호가 할당되며 필요에 따라 계속 업데이트됩니다. PIP* 패키지 관리 유틸리티를 사용하여 사용자 환경에 패키지를 설치해야 합니다.

astraSDK.py 클래스를 설치한 후에는 를 배치함으로써 활용할 수 있습니다 `import astraSDK` 있습니다. 또한, `actoolkit` 는 명령 프롬프트에서 직접 호출할 수 있으며 와 동일합니다 `toolkit.py` (`actoolkit list clusters` 은 과 동일합니다 `./toolkit.py list clusters`)를 클릭합니다.

을 참조하십시오 "PyPI: NetApp Astra Control Python SDK" 를 참조하십시오.

#### GitHub 소스 코드입니다

SDK 소스 코드는 GitHub에서도 사용할 수 있습니다. 리포지토리에는 다음이 포함됩니다.

- `astraSDK.py` (Python 수업이 포함된 Astra SDK)
- `toolkit.py` (상위 수준 기능 기반 스크립트)
- 자세한 설치 요구 사항 및 지침
- 설치 스크립트
- 추가 문서

를 클론 복제할 수 있습니다 "GitHub:NetApp/NetApp-Astra-툴킷" 로컬 환경에 리포지토리.

## 설치 및 기본 요구 사항

패키지 설치 및 사용 준비 과정에서 고려해야 할 몇 가지 옵션과 요구 사항이 있습니다.

### 설치 옵션 요약

다음 방법 중 하나로 SDK를 설치할 수 있습니다.

- 준비된 을 사용합니다 "Docker:NetApp/Astra-툴킷" 이미지 - 를 포함하여 필요한 모든 종속 항목이 설치되어 있습니다 `actoolkit`
- Pip을 사용하여 를 설치합니다 `actoolkit` PyPI에서 Python 환경으로 패키지를 만듭니다
- GitHub 리포지토리를 복제하고 Python 클라이언트 코드에 액세스할 수 있도록 두 개의 핵심 Python 파일을 복사/수정합니다

자세한 내용은 PyPI 및 GitHub 페이지를 참조하십시오.

### **Astra Control** 환경에 대한 요구 사항

Astra SDK의 Python 클래스를 직접 사용하는지 또는 의 함수를 사용하는지 여부 `toolkit.py` 스크립트 - 궁극적으로 Astra Control 구축 시 REST API에 액세스합니다. 이 때문에 API 토큰과 함께 Astra 계정이 필요합니다. 을 참조하십시오 "시작하기 전에" 자세한 내용은 이 설명서의 * 시작하기 * 섹션의 다른 페이지를 참조하십시오.

### **NetApp Astra Control Python SDK**의 요구사항

SDK에는 로컬 Python 환경과 관련된 몇 가지 사전 요구 사항이 있습니다. 예를 들어 Python 3.8 이상을 사용해야 합니다. 또한 필요한 여러 가지 Python 패키지가 있습니다. 자세한 내용은 GitHub 리포지토리 페이지 또는 PyPI 패키지 페이지를 참조하십시오.

## 유용한 리소스 요약

다음은 시작하는 데 필요한 몇 가지 리소스입니다.

- "PyPI: NetApp Astra Control Python SDK"
- "GitHub:NetApp/NetApp-Astra-툴킷"
- "Docker:NetApp/Astra-툴킷"

# 기본 Python

## 시작하기 전에

Python은 데이터 센터 자동화를 위한 인기 있는 개발 언어입니다. Python의 기본 기능과 여러 가지 공통 패키지를 함께 사용하기 전에 환경 및 필요한 입력 파일을 준비해야 합니다.

Python을 사용하여 Astra Control REST API에 직접 액세스하는 것 외에도 NetApp은 API를 추상화하고 일부 복잡성을 제거하는 툴킷 패키지도 제공합니다. 을 참조하십시오 "NetApp Astra Control Python SDK" 를 참조하십시오.

### 환경을 준비합니다

Python 스크립트를 실행하기 위한 기본 구성 요구 사항은 아래에 설명되어 있습니다.

**Python 3**

최신 버전의 Python 3이 설치되어 있어야 합니다.

#### 추가 라이브러리

Requests* 및 * urllllib3* 라이브러리가 설치되어 있어야 합니다. PIP 또는 다른 Python 관리 도구를 환경에 맞게 사용할 수 있습니다.

#### 네트워크 액세스

스크립트가 실행되는 워크스테이션에는 네트워크 액세스 권한이 있어야 하며 Astra Control에 연결할 수 있어야 합니다. Astra Control Service를 사용할 때는 인터넷에 연결되어 있어야 하며 에 있는 서비스에 연결할 수 있어야 합니다 `https://astra.netapp.io`.

**ID** 정보

계정 ID와 API 토큰을 가진 유효한 Astra 계정이 필요합니다. 을 참조하십시오 "API 토큰을 가져옵니다" 를 참조하십시오.

### JSON 입력 파일을 생성합니다

Python 스크립트는 JSON 입력 파일에 포함된 구성 정보에 의존합니다. 샘플 파일은 아래에 제공됩니다.

환경에 맞게 샘플을 업데이트해야 합니다.

**ID** 정보

다음 파일에는 API 토큰과 Astra 계정이 포함되어 있습니다. 를 사용하여 이 파일을 Python 스크립트로 전달해야 합니다 `-i` (또는 `--identity`) CLI 매개 변수입니다.

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

## 앱을 나열합니다

다음 스크립트를 사용하여 Astra 계정의 응용 프로그램을 나열할 수 있습니다.

을 참조하십시오 "시작하기 전에" 필요한 JSON 입력 파일의 예

```
#!/usr/bin/env python3
##-----------------------------------------------------------------------------
------
#
# Usage: python3 list_man_apps.py -i identity_file.json
#
# (C) Copyright 2022 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 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/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
    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 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)
```

저작권 정보

Copyright © 2025 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.