Astra Control Automation 22.04 문서

Astra Automation 22.04

NetApp
December 04, 2023

목차

Astra Control Automation 22.04 문서

릴리스 정보

이 릴리스에 대해

이 사이트의 문서에서는 Astra Control REST API 및 Astra Control의 2022(22.04) 릴리스에서 사용할 수 있는 관련 자동화 기술에 대해 설명합니다. 특히 이 REST API 릴리스는 Astra Control Center 및 Astra Control Service의 해당 22.04 릴리스에 포함되어 있습니다.

이 릴리스와 이전 릴리스에 대한 자세한 내용은 다음 페이지 및 사이트를 참조하십시오.

- "Astra Control REST API의 새로운 기능"
- "REST 리소스 및 엔드포인트"
- "Astra Control Center 22.04 문서"
- "Astra Control Service 문서"
- "이전 버전의 Astra Automation 설명서"

Astra Control REST API의 새로운 기능

NetApp은 새로운 기능, 개선 사항 및 버그 수정을 제공하기 위해 Astra Control REST API를 정기적으로 업데이트합니다.

2022년 4월 26일(22.04)

이 릴리스는 REST API의 확장 및 업데이트뿐 아니라 향상된 보안 및 관리 기능도 포함합니다.

새로운 고급 **Astra** 리소스

패키지 * 와 * 업그레이드 * 의 두 가지 새로운 리소스 유형이 추가되었습니다. 또한 여러 기존 리소스의 버전이 업그레이드되었습니다.

네임스페이스 세분화를 통해 **RBAC** 강화

연결된 사용자에게 역할을 바인딩하는 경우 사용자가 액세스할 수 있는 네임스페이스를 제한할 수 있습니다. Role Binding API * 참조 및 를 참조하십시오 "RBAC 보안" 를 참조하십시오.

버킷 제거

버킷이 더 이상 필요하지 않거나 제대로 작동하지 않을 경우 이를 제거할 수 있습니다.

Cloud Volumes ONTAP 지원

이제 Cloud Volumes ONTAP가 스토리지 백엔드로 지원됩니다.

추가 제품 개선 사항

두 Astra Control 제품 구현에 대한 몇 가지 추가 개선 사항은 다음과 같습니다.

- Astra Control Center의 일반적인 침투입니다
- AKS의 프라이빗 클러스터
- Kubernetes 1.22 지원
- VMware Tanzu 포트폴리오 지원

Astra Control Center 및 Astra Control Service 문서 사이트에서 * 새로운 기능 * 페이지를 참조하십시오.

관련 정보

- "Astra Control Center: 새로운 기능"
- "Astra Control Service: 새로운 기능"

2021년 12월 14일(21.12)

이 릴리스에는 향후 릴리즈 업데이트를 통해 Astra Control의 발전을 더욱 잘 지원하기 위해 문서 구조의 변경 사항과 함께 REST API 확장이 포함되어 있습니다.

Astra Control의 각 릴리스에 대한 Astra 자동화 문서를 분리합니다

Astra Control의 모든 릴리스에는 특정 릴리스의 기능에 맞게 향상되고 조정된 고유한 REST API가 포함되어 있습니다. Astra Control REST API의 각 릴리스에 대한 문서는 이제 관련 GitHub 콘텐츠 저장소와 함께 자체 전용 웹 사이트에서 제공됩니다. 주 문서 사이트입니다 "Astra 제어 자동화" 항상 최신 릴리스에 대한 설명서를 포함합니다. 을 참조하십시오 "이전 버전의 Astra Control Automation 설명서" 이전 릴리즈에 대한 자세한 내용은.

REST 리소스 유형의 확장

REST 리소스 유형의 수는 실행 후크와 스토리지 백엔드에 중점을 두고 계속 확장됩니다. 새로운 리소스에는 계정, 실행 후크, 후크 소스, 실행 후크 재정의, 클러스터 노드, 관리 스토리지 백엔드, 네임스페이스, 스토리지 디바이스 및 스토리지 노드 을 참조하십시오 "리소스" 를 참조하십시오.

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK는 오픈 소스 패키지로서, Astra Control 환경을 위한 자동화 코드를 더욱 쉽게 개발할 수 있도록 지원합니다. 핵심 요소는 REST API 호출의 복잡성을 추상화하는 일련의 클래스가 포함된 Astra SDK입니다. 또한 Python 클래스를 래핑하고 추상화하여 특정 관리 작업을 실행하는 툴킷 스크립트가 있습니다. 을 참조하십시오 "NetApp Astra Control Python SDK" 를 참조하십시오.

2021년 8월 5일(21.08)

이 릴리스에는 새로운 Astra 배포 모델 및 REST API의 주요 확장이 포함되어 있습니다.

Astra Control Center 배포 모델

이 릴리즈에는 퍼블릭 클라우드 서비스로 제공되는 기존 Astra Control Service 오퍼링 외에도 Astra Control Center 온프레미스 구축 모델도 포함되어 있습니다. 사이트에 Astra Control Center를 설치하여 로컬 Kubernetes 환경을 관리할 수 있습니다. 두 Astra Control 배포 모델은 동일한 REST API를 공유하며, 설명서에 명시된 바와 같이 약간의 차이가 있습니다.

REST 리소스 유형의 확장

Astra Control REST API를 통해 액세스할 수 있는 리소스의 수가 크게 확장되었으며, 새로운 리소스 중 다수가 사내 Astra Control Center 오퍼링을 위한 기반을 제공하고 있습니다. 새로운 리소스에는 ASUP, 권한, 기능, 라이센스, 설정, 가입형, 버킷, 클라우드, 클러스터, 관리 클러스터, 스토리지 백엔드 및 스토리지 클래스 을 참조하십시오 "리소스" 를 참조하십시오.

Astra 구축을 지원하는 추가 엔드포인트에는

확장된 REST 리소스 외에도 Astra Control 구축을 지원하는 데 사용할 수 있는 여러 가지 새로운 API 엔드포인트가 있습니다.

OpenAPI 지원

OpenAPI 엔드포인트는 현재 OpenAPI JSON 문서 및 기타 관련 리소스에 대한 액세스를 제공합니다.

OpenMetrics 지원

OpenMetrics 엔드포인트는 OpenMetrics 리소스를 통해 계정 메트릭에 액세스할 수 있습니다.

2021년 4월 15일(21.04)

이 릴리즈에는 다음과 같은 새로운 기능과 향상된 기능이 포함되어 있습니다.

REST API 소개

Astra Control REST API는 Astra Control Service 오퍼링과 함께 사용할 수 있습니다. REST 기술과 현재의 모범 사례를 기반으로 개발되었으며 API는 Astra 구축을 자동화하기 위한 기반을 제공하며 다음과 같은 기능과 이점을 제공합니다.

리소스

14개의 REST 리소스 유형을 사용할 수 있습니다.

API 토큰 액세스

REST API에 대한 액세스는 Astra 웹 사용자 인터페이스에서 생성할 수 있는 API 액세스 토큰을 통해 제공됩니다. API 토큰은 API에 대한 보안 액세스를 제공합니다.

컬렉션 지원

리소스 컬렉션에 액세스하는 데 사용할 수 있는 다양한 쿼리 매개 변수 집합이 있습니다. 지원되는 일부 작업에는 필터링, 정렬 및 페이지 매김이 포함됩니다.

알려진 문제

Astra Control REST API와 관련된 현재 릴리즈에 대해 알려진 모든 문제를 검토해야 합니다. 알려진 문제는 제품을 성공적으로 사용하지 못하게 만들 수 있는 문제를 식별합니다.

Astra Control REST API 22.04 릴리스에는 새로 알려진 문제가 없습니다. 아래 설명된 문제는 이전 릴리즈에서 발견되었으며 현재 릴리스에서도 계속 적용됩니다.

백엔드 스토리지 노드의 모든 스토리지 디바이스가 검색되지 않습니다

스토리지 노드에 정의된 스토리지 디바이스를 검색하기 위해 REST API 호출을 실행하는 경우 일부 디바이스가 반환되지 않습니다.

Astra Control REST API 소개

Astra Control Center 및 Astra Control Service는 공통 REST API를 제공하여 Curl과 같은 프로그래밍 언어 또는 유틸리티를 통해 직접 액세스할 수 있습니다. API의 주요 특징 및 이점은 다음과 같습니다.

> REST API에 액세스하려면 먼저 Astra 웹 사용자 인터페이스에 로그인하고 API 토큰을 생성해야 합니다. 각 API 요청에 토큰을 포함해야 합니다.

REST 기술 기반

Astra Control API는 REST 기술과 최신 Best Practice를 사용하여 만들어졌습니다. 코어 기술에는 HTTP, JSON 및 RBAC가 포함됩니다.

Astra Control 배포 모델 **2**개 지원

Astra Control Service는 퍼블릭 클라우드 환경 내에서 사용되고, Astra Control Center는 사내 배포용으로 사용됩니다. 이러한 구축 모델을 모두 지원하는 하나의 REST API가 있습니다.

REST 엔드포인트 리소스와 객체 모델 간의 매핑을 해제합니다

리소스 맵에 액세스하는 데 사용되는 외부 REST 엔드포인트는 Astra 서비스에서 내부적으로 유지 관리하는 일관된 오브젝트 모델에 있습니다. 객체 모델은 API 작업과 응답을 명확하게 정의하는 데 도움이 되는 ER(Entity-relationship) 모델링을 사용하여 설계되었습니다.

다양한 쿼리 매개 변수 집합

REST API는 리소스 컬렉션에 액세스하는 데 사용할 수 있는 풍부한 쿼리 매개 변수 집합을 제공합니다. 지원되는 일부 작업에는 필터링, 정렬 및 페이지 매김이 포함됩니다.

Astra Control 웹 **UI**와 정렬

Astra 웹 사용자 인터페이스의 설계는 REST API와 일치하므로 두 액세스 경로와 사용자 경험 간에 일관성이 있습니다.

강력한 디버깅 및 문제 확인 데이터

Astra Control REST API는 시스템 이벤트 및 사용자 알림을 포함하여 강력한 디버깅 및 문제 확인 기능을 제공합니다.

워크플로 프로세스

자동화 코드 개발을 지원하기 위해 일련의 워크플로우가 제공됩니다. 워크플로우는 인프라 및 관리의 두 가지 범주로 구성됩니다.

고급 자동화 기술의 기반

REST API에 직접 액세스하는 것 외에도 REST API를 기반으로 하는 다른 자동화 기술을 사용할 수 있습니다.

Astra 제품군 문서의 일부입니다

Astra Control Automation 문서는 대규모 Astra 제품군 문서의 일부입니다. 을 참조하십시오 "Astra 문서" 를 참조하십시오.

시작하십시오

시작하기 전에

아래 단계를 검토하여 Astra Control REST API를 시작할 수 있도록 빠르게 준비할 수 있습니다.

Astra 계정 자격 증명을 가지고 있어야 합니다

Astra 웹 사용자 인터페이스에 로그인하고 API 토큰을 생성하려면 Astra 자격 증명이 필요합니다. Astra Control Center를 사용하면 이러한 자격 증명을 로컬로 관리할 수 있습니다. Astra Control Service를 사용하면 * Auth0 * 서비스를 통해 계정 자격 증명에 액세스할 수 있습니다.

Kubernetes의 기본 개념에 대해 잘 알고 있어야 합니다

Kubernetes의 몇 가지 기본 개념에 익숙해야 합니다. 을 참조하십시오 "기본 Kubernetes 개념" 를 참조하십시오.

REST 개념 및 구현을 검토합니다

반드시 검토하십시오 "핵심 REST 구현" REST 개념과 Astra Control REST API의 설계 방식에 대한 자세한 내용은

자세한 정보를 확인하십시오

에 나와 있는 추가 정보 리소스를 숙지해야 합니다 "추가 리소스".

API 토큰을 가져옵니다

Astra Control REST API를 사용하려면 Astra API 토큰을 얻어야 합니다.

소개

API 토큰은 Astra에 호출자를 식별하며 모든 REST API 호출에 포함되어야 합니다.

- Astra 웹 사용자 인터페이스를 사용하여 API 토큰을 생성할 수 있습니다.
- 토큰과 함께 전달된 사용자 ID는 토큰을 생성하는 사용자에 의해 결정됩니다.
- 토큰은 인가 HTTP 요청 헤더에 포함되어야 합니다.
- 토큰이 생성된 후에는 만료되지 않습니다.
- Astra 웹 사용자 인터페이스에서 토큰을 취소할 수 있습니다.

관련 정보

- "API 토큰을 취소합니다"

Astra API 토큰을 생성합니다

다음 단계에서는 Astra API 토큰을 생성하는 방법을 설명합니다.

시작하기 전에

Astra 계정에 대한 자격 증명이 필요합니다.

이 작업에 대해

이 작업은 Astra 웹 인터페이스에서 API 토큰을 생성합니다. 또한 API 호출 시 필요한 계정 ID도 검색해야 합니다.

단계

1. 계정 자격 증명을 사용하여 Astra에 로그인합니다.

   Astra Control Service에 대한 다음 사이트에 액세스합니다. "https://astra.netapp.io"

2. 페이지 오른쪽 상단의 그림 아이콘을 클릭하고 * API access * 를 선택합니다.
3. 페이지에서 * API 토큰 생성 * 을 클릭하고 팝업 창에서 * API 토큰 생성 * 을 클릭합니다.
4. 아이콘을 클릭하여 토큰 문자열을 클립보드에 복사하고 편집기에 저장합니다.
5. 동일한 페이지에서 사용할 수 있는 계정 ID를 복사하여 저장합니다.

작업을 마친 후

Curl 또는 프로그래밍 언어를 통해 Astra Control REST API에 액세스할 때는 HTTP 'Authorization' 요청 헤더에 API bearer 토큰을 포함시켜야 합니다.

헬로우 월드

워크스테이션 CLI에서 간단한 Curl 명령을 실행하여 Astra Control REST API를 사용하여 사용 가능 여부를 확인할 수 있습니다.

시작하기 전에

Curl 유틸리티는 로컬 워크스테이션에서 사용할 수 있어야 합니다. 또한 API 토큰과 관련 계정 식별자가 있어야 합니다. 을 참조하십시오 "API 토큰을 가져옵니다" 를 참조하십시오.

컬의 예

다음 Curl 명령은 Astra 사용자 목록을 검색합니다. 표시된 대로 적절한 <account_ID> 및 <api_token>을 제공합니다.

```
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 출력 예

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

워크플로우 사용을 준비하십시오

실제 배포와 함께 사용하기 전에 Astra 워크플로의 구성 및 형식을 숙지해야 합니다.

소개

Workflow _ 은(는) 특정 관리 작업 또는 목표를 달성하는 데 필요한 하나 이상의 단계 시퀀스입니다. Astra Control 워크플로우의 각 단계는 다음 중 하나입니다.

- REST API 호출(curl 및 JSON 예 등의 세부 정보 포함)
- 다른 Astra 워크플로 호출
- 기타 관련 작업(예: 필요한 설계 결정)

워크플로에는 각 작업을 수행하는 데 필요한 핵심 단계와 매개 변수가 포함됩니다. 또한 자동화 환경을 사용자 지정할 수 있는 출발점을 제공합니다.

공통 입력 매개변수

아래에 설명된 입력 매개 변수는 REST API 호출을 나타내는 데 사용되는 모든 curl 샘플에 공통으로 사용됩니다.

> 이러한 입력 매개 변수는 보편적으로 필요하기 때문에 개별 워크플로에서 더 이상 설명되지 않습니다. 특정 curl 예제에 추가 입력 매개 변수가 사용되는 경우 * 추가 입력 매개 변수 * 절에 설명되어 있습니다.

경로 매개 변수

모든 REST API 호출에 사용되는 엔드포인트 경로에는 다음 매개 변수가 포함됩니다. 도 참조하십시오 "URL 형식" 를 참조하십시오.

계정 **ID**입니다

API 작업이 실행되는 Astra 계정을 식별하는 UIDv4 값이다. 을 참조하십시오 "API 토큰을 가져옵니다" 계정 ID를 찾는 방법에 대한 자세한 내용은 를 참조하십시오.

요청 헤더

REST API 호출에 따라 몇 가지 요청 헤더를 포함해야 할 수도 있습니다.

권한 부여

워크플로의 모든 API 호출에는 사용자를 식별하기 위한 API 토큰이 필요합니다. '권한 부여' 요청 헤더에 토큰을 포함해야 합니다. 을 참조하십시오 "API 토큰을 가져옵니다" API 토큰 생성에 대한 자세한 내용은 를 참조하십시오.

콘텐츠 유형

HTTP POST 및 PUT 요청을 사용하여 요청 본문에 JSON이 포함된 경우 Astra 리소스를 기반으로 미디어 유형을 선언해야 합니다. 예를 들어, 관리 대상 애플리케이션에 대한 스냅샷을 생성할 때 "Content-Type:application/astra-appSnap+json" 머리글을 포함할 수 있습니다.

수락

Astra 리소스를 기반으로 응답에서 예상하는 콘텐츠의 특정 미디어 유형을 선언할 수 있습니다. 예를 들어, 관리 대상 애플리케이션의 백업을 나열할 때 "Accept:application/astra-appBackup+json" 머리글을 포함할 수 있습니다. 그러나 간단히 하기 위해 워크플로의 curl 샘플에는 모든 미디어 유형이 적용됩니다.

토큰 및 식별자 표시

curl 예제에 사용된 API 토큰과 기타 ID 값은 식별 가능한 의미 없이 불투명합니다. 따라서 샘플의 가독성을 향상시키기 위해 실제 토큰 및 ID 값은 사용되지 않습니다. 대신, 다음과 같은 몇 가지 이점을 제공하는 더 작은 예약 키워드가 사용됩니다.

- CURL 및 JSON 샘플은 보다 명확하고 이해하기 쉽습니다.
- 모든 키워드는 대괄호와 대문자를 사용하여 동일한 형식을 사용하기 때문에 삽입하거나 추출할 위치와 내용을 빠르게 식별할 수 있습니다.
- 원래 매개 변수를 복사하여 실제 배포에서 사용할 수 없으므로 값이 손실되지 않습니다.

다음은 curl 예제에 사용되는 몇 가지 일반적인 예약된 키워드입니다. 이 목록은 전체 목록이 아니며 필요에 따라 추가 키워드가 사용됩니다. 그들의 의미는 상황에 따라 분명해야 합니다.

키워드	유형	설명
account_ID>입니다	경로	API 작업이 실행되는 계정을 식별하는 UIDv4 값입니다.
api_token>	머리글	호출자를 식별하고 승인하는 베어러 토큰.
managed_app_ID> 를 선택합니다	경로	API 호출에 대한 관리되는 응용 프로그램을 식별하는 UIDv4 값입니다.

워크플로 범주

배포 모델에 따라 두 가지 범주의 Astra 워크플로를 사용할 수 있습니다. Astra Control Center를 사용하는 경우 인프라 워크플로로 시작한 다음 관리 워크플로로 넘어가야 합니다. Astra Control Service를 사용하는 경우 일반적으로 관리 워크플로로 직접 이동할 수 있습니다.

> 워크플로의 curl 샘플은 Astra Control Service의 URL을 사용합니다. 사내 Astra Control Center를 사용하는 경우 해당 환경에 맞게 URL을 변경해야 합니다.

인프라 워크플로우

이러한 워크플로우에서는 자격 증명, 버킷 및 스토리지 백엔드를 포함한 Astra 인프라를 처리합니다. Astra Control Center와 함께 필요하지만 대부분의 경우 Astra Control Service와 함께 사용할 수 있습니다. 워크플로우는 Astra 관리 클러스터를 설정하고 유지하는 데 필요한 작업에 중점을 둡니다.

관리 워크플로

관리되는 클러스터가 있는 경우 이러한 워크플로우를 사용할 수 있습니다. 워크플로우는 애플리케이션 보호 및 관리 앱 백업, 복원, 클론 복제 등의 지원 작업에 중점을 둡니다.

기본 **Kubernetes** 개념

Astra REST API를 사용할 때는 몇 가지 Kubernetes 개념을 이해해야 합니다.

오브젝트

Kubernetes 환경 내에 유지되는 객체는 클러스터 구성을 나타내는 영구 엔터티입니다. 이러한 오브젝트는 클러스터 워크로드를 포함한 시스템의 상태를 종합적으로 설명합니다.

네임스페이스

네임스페이스는 단일 클러스터 내에서 리소스를 격리하는 기술을 제공합니다. 이 조직 구조는 작업, 사용자 및 자원의 유형을 나눌 때 유용합니다. _namespace scope_의 개체는 네임스페이스 내에서 고유해야 하지만 _cluster scope_를 가진 개체는 전체 클러스터에서 고유해야 합니다.

라벨

라벨은 Kubernetes 오브젝트와 연결될 수 있습니다. 이들은 키 값 쌍을 사용하여 특성을 설명하고, 조직에 유용할 수 있지만 핵심 Kubernetes 운영 이외의 클러스터에 임의 조직을 적용할 수 있습니다.

핵심 REST 구현

REST 웹 서비스

REST(Representational State Transfer)는 분산된 웹 애플리케이션을 만드는 스타일입니다. 웹 서비스 API 설계에 적용할 경우 서버 기반 리소스를 노출하고 상태를 관리하기 위한 주요 기술 및 모범 사례 집합을 설정합니다. REST는 애플리케이션 개발을 위한 일관된 기반을 제공하지만 각 API의 세부 사항은 특정 설계 선택에 따라 달라질 수 있습니다. 라이브 구축과 함께 사용하기 전에 Astra Control REST API의 특성을 알고 있어야 합니다.

리소스 및 상태 표시

리소스는 웹 기반 시스템의 기본 구성 요소입니다. REST 웹 서비스 응용 프로그램을 만들 때 초기 설계 작업은 다음과 같습니다.

- 시스템 또는 서버 기반 리소스 식별

  모든 시스템은 리소스를 사용하고 유지합니다. 리소스는 파일, 비즈니스 트랜잭션, 프로세스 또는 관리 엔티티가 될 수 있습니다. REST 웹 서비스를 기반으로 애플리케이션을 설계하는 첫 번째 작업 중 하나는 리소스를 식별하는 것입니다.

- 자원 상태 및 연관된 상태 작업의 정의

  리소스는 항상 한정된 수의 상태 중 하나에 있습니다. 상태 변경에 영향을 주는 데 사용되는 상태 및 관련 작업을 명확하게 정의해야 합니다.

URI 끝점

모든 REST 리소스는 잘 정의된 주소 지정 체계를 사용하여 정의되고 사용 가능해야 합니다. 리소스가 있고 식별된 끝점은 URI(Uniform Resource Identifier)를 사용합니다. URI는 네트워크의 각 리소스에 대해 고유한 이름을 만들기 위한 일반 프레임워크를 제공합니다. URL(Uniform Resource Locator)은 리소스를 식별하고 액세스하기 위해 웹 서비스와 함께 사용되는 URI 유형입니다. 일반적으로 리소스는 파일 디렉터리와 비슷한 계층적 구조로 표시됩니다.

HTTP 메시지

HTTP(Hypertext Transfer Protocol)는 웹 서비스 클라이언트 및 서버가 리소스에 대한 요청 및 응답 메시지를 교환하기 위해 사용하는 프로토콜입니다. 웹 서비스 응용 프로그램 설계의 일환으로 HTTP 메서드는 리소스 및 해당 상태 관리 작업에 매핑됩니다. HTTP는 상태 비저장 입니다. 따라서 관련 요청 및 응답 집합을 하나의 트랜잭션으로 연결하려면 요청 및 응답 데이터 플로우와 함께 전달된 HTTP 헤더에 추가 정보가 포함되어야 합니다.

JSON 형식

정보는 여러 가지 방법으로 웹 서비스 클라이언트와 서버 간에 구조화되고 전송될 수 있지만 가장 일반적인 옵션은 JSON(JavaScript Object Notation)입니다. JSON은 단순 데이터 구조를 일반 텍스트로 나타내는 업계 표준이며 리소스를 설명하는 상태 정보를 전송하는 데 사용됩니다. Astra Control REST API는 JSON을 사용하여 각 HTTP 요청 및 응답의 본문으로 전송되는 데이터를 포맷합니다.

리소스 및 컬렉션

Astra Control REST API는 리소스 인스턴스 및 리소스 인스턴스 컬렉션에 대한 액세스를 제공합니다.

> 개념적으로 REST * 리소스 * 는 OOP(개체 지향 프로그래밍) 언어 및 시스템에 정의된 * 개체 * 와 유사합니다. 때때로 이러한 용어는 서로 바꿔서 사용할 수 있습니다. 그러나 일반적으로 "리소스"는 외부 REST API의 컨텍스트에서 사용하는 반면 "개체"는 서버에 저장된 해당 상태 저장 인스턴스 데이터에 사용됩니다.

Astra 리소스의 속성

Astra Control REST API는 RESTful 설계 원칙을 준수합니다. 각 Astra 리소스 인스턴스는 잘 정의된 리소스 유형을 기반으로 생성됩니다. 같은 형식의 리소스 인스턴스 집합을 * 컬렉션 * 이라고 합니다. API 호출은 개별 리소스 또는 리소스 모음에 대해 작동합니다.

리소스 유형

Astra Control REST API에 포함된 리소스 유형은 다음과 같은 특성을 갖습니다.

- 모든 리소스 유형은 스키마(일반적으로 JSON)를 사용하여 정의됩니다.
- 모든 리소스 스키마에는 리소스 유형과 버전이 포함됩니다
- 리소스 유형은 전역적으로 고유합니다

리소스 인스턴스

Astra Control REST API를 통해 사용 가능한 리소스 인스턴스의 특징은 다음과 같습니다.

- 리소스 인스턴스는 단일 리소스 유형에 따라 만들어집니다
- 리소스 유형은 미디어 유형 값을 사용하여 표시됩니다
- 인스턴스는 Astra 서비스에서 유지 관리하는 상태 저장 데이터로 구성됩니다
- 각 인스턴스는 고유하고 오래 지속되는 URL을 통해 액세스할 수 있습니다
- 리소스 인스턴스에 둘 이상의 표현이 있을 수 있는 경우 다양한 미디어 유형을 사용하여 원하는 표현을 요청할 수 있습니다

리소스 컬렉션

Astra Control REST API를 통해 사용 가능한 리소스 수집에는 다음과 같은 특성이 있습니다.

- 단일 리소스 형식의 리소스 인스턴스 집합을 컬렉션이라고 합니다
- 리소스 컬렉션에는 고유하고 오래 지속되는 URL이 있습니다

인스턴스 식별자

모든 리소스 인스턴스는 만들 때 식별자가 할당됩니다. 이 식별자는 128비트 UIDv4 값입니다. 할당된 UUIDv4 값은 전역적으로 고유하며 변경할 수 없습니다. 새 인스턴스를 만드는 API 호출을 실행하면 관련 ID가 있는 URL이 HTTP 응답의 "위치" 헤더에서 호출자에게 반환됩니다. 식별자를 추출하여 리소스 인스턴스를 참조할 때 후속 호출에 사용할 수 있습니다.

리소스 식별자는 컬렉션에 사용되는 기본 키입니다.

Astra 리소스의 공통 구조

모든 Astra Control 리소스는 공통 구조를 사용하여 정의됩니다.

공통 데이터

모든 Astra 리소스에는 다음 표에 나와 있는 키 값이 포함되어 있습니다.

키	설명
유형	자원 유형 * 이라고 하는 전역적으로 고유한 자원 유형입니다.
버전	리소스 버전*이라고 하는 버전 식별자입니다.
ID입니다	리소스 식별자 * 라고 하는 전역 고유 식별자입니다.
메타데이터	사용자 및 시스템 레이블을 비롯한 다양한 정보를 포함하는 JSON 개체입니다.

메타데이터 개체입니다

각 Astra 리소스에 포함된 메타데이터 JSON 개체에는 다음 표에 나와 있는 키 값이 포함됩니다.

키	설명
라벨	리소스와 연결된 클라이언트 지정 레이블의 JSON 배열입니다.
CreationTimestamp 를 클릭합니다	JSON 문자열에는 리소스가 생성된 시점을 나타내는 타임스탬프가 포함되어 있습니다.
modificationTimestamp	ISO-8601 형식의 타임 스탬프가 포함된 JSON 문자열로, 리소스가 마지막으로 변경된 시기를 나타냅니다.
생성 시	리소스를 생성한 사용자 ID의 UUIDv4 식별자가 포함된 JSON 문자열입니다. 내부 시스템 구성 요소에 의해 리소스가 생성되었고 생성 엔티티와 연관된 UUID가 없는 경우 * null * UUID가 사용됩니다.

리소스 상태입니다

선택한 리소스 수명 주기 전환을 조정하고 액세스를 제어하는 데 사용되는 '상태' 값입니다.

HTTP 세부 정보

Astra Control REST API는 HTTP 및 관련 매개 변수를 사용하여 리소스 및 컬렉션에 대한 작업을 수행합니다. HTTP 구현에 대한 자세한 내용은 아래에 나와 있습니다.

API 트랜잭션 및 CRUD 모델

Astra Control REST API는 잘 정의된 작업과 상태 전환을 통해 트랜잭션 모델을 구현합니다.

요청 및 응답 **API** 트랜잭션

모든 REST API 호출은 Astra 서비스에 대한 HTTP 요청으로 수행됩니다. 각 요청은 연결된 응답을 클라이언트에 다시 생성합니다. 이 요청 응답 쌍은 API 트랜잭션으로 간주될 수 있습니다.

CRUD 운영 모델 지원

Astra Control REST API를 통해 사용 가능한 각 리소스 인스턴스 및 컬렉션은 * CRUD * 모델을 기반으로 액세스합니다. 4개의 작업이 있으며, 각 작업은 단일 HTTP 메서드에 매핑됩니다. 다음과 같은 작업이 포함됩니다.

- 생성
- 읽기
- 업데이트
- 삭제

일부 Astra 리소스의 경우 이러한 작업의 일부만 지원됩니다. 를 검토해야 합니다 "API 참조입니다" 특정 API 호출에 대한 자세한 내용은 를 참조하십시오.

HTTP 메서드

API에서 지원하는 HTTP 메서드 또는 동사는 아래 표에 나와 있습니다.

방법	**CRUD**	설명
가져오기	읽기	리소스 인스턴스 또는 컬렉션의 개체 속성을 검색합니다. 이 작업은 컬렉션과 함께 사용할 때 * list * 연산으로 간주됩니다.
게시	생성	입력 매개 변수를 기반으로 새 리소스 인스턴스를 만듭니다. 장기간의 URL은 Location(위치) 응답 헤더로 반환됩니다.
를 누릅니다	업데이트	제공된 JSON 요청 본문으로 전체 리소스 인스턴스를 업데이트합니다. 사용자가 수정할 수 없는 키 값은 보존됩니다.
삭제	삭제	기존 리소스 인스턴스를 삭제합니다.

요청 및 응답 헤더

다음 표는 Astra Control REST API와 함께 사용되는 HTTP 헤더를 요약한 것입니다.

을 참조하십시오 "RFC 7232" 및 "RFC 7233" 를 참조하십시오.

머리글	유형	사용 참고 사항
수락	요청하십시오	값이 " * / * "이거나 제공되지 않으면 Content-Type 응답 헤더에서 application/json이 반환됩니다. 값이 Astra 리소스 미디어 유형으로 설정되어 있으면 Content-Type 헤더에서 동일한 미디어 유형이 반환됩니다.
권한 부여	요청하십시오	사용자에 대한 API 키가 있는 베어러 토큰.
Content-Type(콘텐츠 유형)	응답	'수락' 요청 헤더를 기준으로 반환됩니다.
ETag	응답	RFC 7232에 정의된 대로 에 포함됩니다. 값은 전체 JSON 리소스에 대한 MD5 값의 16진수 표현입니다.
일치하는 경우	요청하십시오	3.1 RFC 7232절에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.

머리글	유형	사용 참고 사항
If-Modified-Since	요청하십시오	섹션 3.4 RFC 7232에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.
수정되지 않은 경우 - 이후	요청하십시오	섹션 3.4 RFC 7232에 설명된 대로 구현된 전제 조건 요청 헤더와 * PUT * 요청 지원.
위치	응답	새로 만든 리소스의 전체 URL을 포함합니다.

쿼리 매개 변수

다음 쿼리 매개 변수를 리소스 모음과 함께 사용할 수 있습니다. 을 참조하십시오 "컬렉션 작업" 를 참조하십시오.

쿼리 매개 변수입니다	설명
포함	컬렉션을 읽을 때 반환되어야 하는 필드를 포함합니다.
필터	컬렉션을 읽을 때 반환할 리소스에 대해 일치해야 하는 필드를 나타냅니다.
주문	컬렉션을 읽을 때 반환되는 리소스의 정렬 순서를 결정합니다.
제한	컬렉션을 읽을 때 반환되는 최대 리소스 수를 제한합니다.
건너뛰기	컬렉션을 읽을 때 전달하고 건너뛸 리소스 수를 설정합니다.
카운트	메타데이터 개체에서 총 리소스 수를 반환해야 하는지 여부를 나타냅니다.

HTTP 상태 코드입니다

Astra Control REST API에서 사용하는 HTTP 상태 코드는 아래와 같다.

> Astra Control REST API는 HTTP API * 표준에 대한 * Problem Details도 사용합니다. 을 참조하십시오 "진단 및 지원" 를 참조하십시오.

코드	의미	설명
200	좋습니다	새 리소스 인스턴스를 만들지 않는 호출의 성공 여부를 나타냅니다.
201	작성됨	객체가 성공적으로 생성되고 위치 응답 헤더에 객체의 고유 식별자가 포함됩니다.
204	콘텐츠가 없습니다	반환된 콘텐츠가 없지만 요청이 성공했습니다.
400	잘못된 요청입니다	요청 입력이 인식되지 않거나 부적절합니다.
401	권한이 없습니다	사용자에게 권한이 없으며 인증이 필요합니다.
403	금지됨	인증 오류로 인해 액세스가 거부되었습니다.
404	찾을 수 없습니다	요청에서 참조되는 리소스가 없습니다.
409	충돌	개체가 이미 있으므로 개체를 만들지 못했습니다.
500입니다	내부 오류입니다	서버에서 일반적인 내부 오류가 발생했습니다.
503	서비스를 사용할 수 없습니다	어떤 이유로 요청을 처리할 준비가 되지 않았습니다.

URL 형식

REST API를 통해 리소스 인스턴스 또는 컬렉션에 액세스하는 데 사용되는 URL의 일반 구조는 여러 값으로 구성됩니다. 이 구조체는 기본 개체 모델 및 시스템 설계를 반영합니다.

root로 계정을 지정합니다

모든 REST 끝점에 대한 리소스 경로의 루트는 Astra 계정입니다. 따라서 URL의 모든 경로는 "/account/{account_id}"로 시작합니다. 여기서 "account_id"는 계정에 대한 고유한 UUIDv4 값입니다. 내부 구조 모든 리소스 액세스가 특정 계정을 기반으로 하는 설계를 반영합니다.

끝점 리소스 범주입니다

Astra 리소스 끝점은 세 가지 범주로 나뉩니다.

- 코어('/코어')
- 관리 애플리케이션("/k8s")
- 토폴로지('/topology')

을 참조하십시오 "리소스" 를 참조하십시오.

카테고리 버전

세 가지 리소스 범주 각각에는 액세스한 리소스의 버전을 제어하는 전역 버전이 있습니다. 규칙과 정의에 따라 새로운 주요 버전의 리소스 범주(예: /v1 에서 / v2)로 이동하면 API의 변경 내용이 깨질 수 있습니다.

리소스 인스턴스 또는 컬렉션입니다

리소스 인스턴스 또는 컬렉션 액세스 여부에 따라 경로에서 리소스 형식과 식별자의 조합을 사용할 수 있습니다.

예

- 리소스 경로입니다

  위에 제시된 구조를 바탕으로 엔드포인트에 대한 일반적인 경로는 "/accounts/{account_id}/core/v1/users"입니다.

- URL을 완료합니다

  해당 끝점의 전체 URL은 'https://astra.netapp.io/accounts/{account_id}/core/v1/users'입니다.

리소스 및 엔드포인트

Astra Control REST API를 통해 제공되는 리소스를 사용하여 Astra 구축을 자동화할 수 있습니다. 각 리소스는 하나 이상의 엔드포인트를 통해 액세스할 수 있습니다. 아래에 제공된 정보는 자동화 배포의 일부로 사용할 수 있는 REST 리소스에 대한 소개를 제공합니다.

> Astra Control 리소스에 액세스하는 데 사용되는 경로 및 전체 URL의 형식은 여러 값을 기반으로 합니다. 을 참조하십시오 "URL 형식" 를 참조하십시오. 도 참조하십시오 "API 참조입니다" Astra 리소스 및 엔드포인트 사용에 대한 자세한 내용은

Astra Control REST 리소스 요약

Astra Control REST API에서 제공하는 기본 리소스 끝점은 세 가지 범주로 구성됩니다. 각 리소스는 명시된 경우를 제외하고 CRUD 작업의 전체 세트(생성, 읽기, 업데이트, 삭제)로 액세스할 수 있습니다.

릴리즈 * 열에는 리소스가 처음 도입되었을 때 Astra 릴리즈가 표시됩니다. 이 필드는 현재 릴리스와 함께 새로 추가된 리소스에 대해 굵게 표시됩니다.

핵심 리소스

핵심 리소스 엔드포인트는 Astra 런타임 환경을 설정하고 유지하는 데 필요한 기본 서비스를 제공합니다.

리소스	놓습니다	설명
계정	21.12	계정 리소스를 사용하여 멀티테넌트 Astra Control 배포 환경 내에서 격리된 테넌트를 관리할 수 있습니다.
ASUP	21.08	ASUP 리소스는 NetApp 지원에 전달된 AutoSupport 번들을 나타냅니다.
자격 증명	21.04	자격 증명 리소스에는 Astra 사용자, 클러스터, 버킷 및 스토리지 백엔드와 함께 사용할 수 있는 보안 관련 정보가 포함되어 있습니다.
소유 권한	21.08	사용 권한 리소스는 활성 라이센스 및 구독에 따라 계정에 사용할 수 있는 기능과 용량을 나타냅니다.
이벤트	21.04	이벤트 리소스는 알림으로 분류된 하위 집합을 포함하여 시스템에서 발생하는 모든 이벤트를 나타냅니다.
실행 후크	21.12	실행 후크 리소스는 관리되는 앱의 스냅샷이 수행되기 전이나 후에 실행할 수 있는 사용자 정의 스크립트를 나타냅니다.
피처	21.08	기능 리소스는 시스템에서 해당 기능이 활성화 또는 비활성화되었는지 확인하기 위해 쿼리할 수 있는 선택한 Astra 기능을 나타냅니다. 액세스는 읽기 전용으로 제한됩니다.
후크 소스	21.12	후크 소스 리소스는 실행 후크와 함께 사용되는 실제 소스 코드를 나타냅니다. 소스 코드를 실행 컨트롤과 분리하면 스크립트를 공유할 수 있도록 하는 등 여러 가지 이점이 있습니다.
라이센스	21.08	라이센스 리소스는 Astra 계정에 사용할 수 있는 라이센스를 나타냅니다.
통지	21.04	알림 리소스는 알림 대상이 있는 Astra 이벤트를 나타냅니다. 액세스 권한은 사용자별로 제공됩니다.

리소스	놓습니다	설명
패키지	* 22.04 *	패키지 리소스는 패키지 정의의 등록 및 액세스를 제공합니다. 소프트웨어 패키지는 파일, 이미지 및 기타 아티팩트를 포함한 다양한 구성 요소로 구성됩니다.
역할 바인딩	21.04	역할 바인딩 리소스는 특정 사용자 쌍과 계정 간의 관계를 나타냅니다. 두 역할 간의 연결 외에도 특정 역할을 통해 각 역할에 대한 권한 집합이 지정됩니다.
설정	21.08	설정 리소스는 특정 Astra 계정의 기능을 설명하는 키-값 쌍의 모음을 나타냅니다.
구독	21.08	서브스크립션 리소스는 Astra 계정에 대한 활성 서브스크립션을 나타냅니다.
토큰	21.04	토큰 리소스는 Astra Control REST API에 프로그래밍 방식으로 액세스하는 데 사용할 수 있는 토큰을 나타냅니다.
읽지 않은 알림	21.04	읽지 않은 알림 리소스는 특정 사용자에게 할당되었지만 아직 읽지 않은 알림을 나타냅니다.
업그레이드	* 22.04 *	업그레이드 리소스를 통해 소프트웨어 구성 요소에 액세스하고 업그레이드를 시작할 수 있습니다.
사용자	21.04	사용자 리소스는 정의된 역할에 따라 시스템에 액세스할 수 있는 Astra 사용자를 나타냅니다.

관리되는 애플리케이션 리소스

관리 애플리케이션 리소스 엔드포인트는 관리되는 Kubernetes 애플리케이션에 대한 액세스를 제공합니다.

리소스	놓습니다	설명
애플리케이션 자산	21.04	애플리케이션 자산 리소스는 Astra 애플리케이션을 관리하는 데 필요한 상태 정보의 내부 컬렉션을 나타냅니다.
애플리케이션 백업	21.04	애플리케이션 백업 리소스는 관리되는 애플리케이션의 백업을 나타냅니다.
애플리케이션 스냅샷	21.04	애플리케이션 스냅샷 리소스는 관리되는 애플리케이션의 스냅샷을 나타냅니다.
실행 후크 재정의	21.12	실행 후크 재정의 리소스를 사용하면 필요에 따라 특정 애플리케이션에 대해 사전 로드된 NetApp 기본 실행 후크를 비활성화할 수 있습니다.
관리형 애플리케이션	21.04	관리 앱 리소스는 Astra에서 관리하는 Kubernetes 애플리케이션을 나타냅니다.
스케줄	21.04	스케줄 리소스는 데이터 보호 정책의 일부로 관리되는 애플리케이션에 대해 예약된 데이터 보호 작업을 나타냅니다.

토폴로지 리소스

토폴로지 리소스 엔드포인트는 관리되지 않는 애플리케이션과 스토리지 리소스에 대한 액세스를 제공합니다.

리소스	놓습니다	설명
애플리케이션	21.04	앱 리소스는 Astra에서 관리하지 않는 애플리케이션을 포함한 모든 Kubernetes 애플리케이션을 나타냅니다.
버킷	21.08	버킷 리소스는 Astra에서 관리하는 애플리케이션의 백업을 저장하는 데 사용되는 S3 클라우드 버킷을 나타냅니다.
클라우드	21.08	클라우드 리소스는 클러스터 및 애플리케이션을 관리하기 위해 Astra 클라이언트가 연결할 수 있는 클라우드를 나타냅니다.

리소스	놓습니다	설명
클러스터	21.08	클러스터 리소스는 Kubernetes에서 관리되지 않는 Kubernetes 클러스터를 나타냅니다.
클러스터 노드	21.12	클러스터 노드 리소스는 Kubernetes 클러스터 내의 개별 노드에 액세스할 수 있도록 허용하여 추가 해결책을 제공합니다.
관리형 클러스터	21.08	관리 클러스터 리소스는 현재 Kubernetes에서 관리되는 Kubernetes 클러스터를 나타냅니다.
관리 스토리지 백엔드	21.12	관리되는 스토리지 백엔드 리소스를 사용하면 백엔드 스토리지 공급자의 추상화된 표현을 액세스할 수 있습니다. 이러한 스토리지 백엔드는 관리형 클러스터와 애플리케이션에서 사용할 수 있습니다.
네임스페이스	21.12	네임스페이스 리소스는 Kubernetes 클러스터 내에서 사용되는 네임스페이스에 대한 액세스를 제공합니다.
스토리지 백엔드	21.08	스토리지 백엔드 리소스는 Astra 관리 클러스터 및 애플리케이션에서 사용할 수 있는 스토리지 서비스 공급자를 나타냅니다.
스토리지 클래스	21.08	스토리지 클래스 리소스는 특정 관리 대상 클러스터에서 발견되어 사용할 수 있는 다양한 클래스 또는 스토리지 유형을 나타냅니다.
볼륨	21.04	볼륨 리소스는 관리 애플리케이션과 관련된 Kubernetes 스토리지 볼륨을 나타냅니다.

현재 릴리즈와 함께 새 엔드포인트를 사용할 수 있습니다

현재 22.04 Astra Control 릴리즈와 함께 다음 REST 끝점이 추가되었습니다. 또한 여러 기존 리소스의 버전이 업그레이드되었습니다.

- /accounts/{account_id}/core/v1/packages
- /accounts/{account_id}/core/v1/packages/{package_id}
- /accounts/{account_id}/core/v1/upgrades
- /accounts/{account_id}/core/v1/upgrades//{upgrade_id}
- /accounts/{account_id}/topology/v1/appBackups
- /accounts/{account_id}/topology/v1/appBackups/{appBackup_id}
- /accounts/{account_id}/topology/v1/cloud/{cloud_id}/cluster/{cluster_id}/clusterNodes
- /accounts/{account_id}/topology/v1/cloud/{cloud_id}/cluster/{cluster_id}/clusterNode/{clusterNode_id}
- /accounts/{account_id}/topology/v1/managedClusters/{managedCluster_id}/apps/{app_id}/appAssets
- /accounts/{account_id}/topology/v1/managedClusters/{managedCluster_id}/apps/{app_id}/appAssets/{appAsset_id}
- /accounts/{account_id}/topology/v1/managedClusters/{managedCluster_id}/clusterNodes
- /accounts/{account_id}/topology/v1/managedClusters/{managedCluster_id}/clusterNode/{clusterNode_id}

추가 리소스 및 엔드포인트

Astra 구축을 지원하는 데 사용할 수 있는 몇 가지 추가 리소스와 엔드포인트가 있습니다.

이러한 리소스 및 엔드포인트는 현재 Astra Control REST API 참조 설명서에 포함되어 있지 않습니다.

OpenAPI를 참조하십시오

OpenAPI 엔드포인트는 현재 OpenAPI JSON 문서 및 기타 관련 리소스에 대한 액세스를 제공합니다.

OpenMetrics

OpenMetrics 엔드포인트는 OpenMetrics 리소스를 통해 계정 메트릭에 액세스할 수 있도록 합니다. Astra Control Center 배포 모델을 통해 지원을 받을 수 있습니다.

추가 사용 고려 사항

RBAC 보안

Astra REST API는 역할 기반 액세스 제어(RBAC)를 지원하여 시스템 기능에 대한 액세스를 제한합니다.

아스트라 역할

모든 Astra 사용자는 수행할 수 있는 작업을 결정하는 단일 역할에 할당됩니다. 역할은 아래 표에 설명된 대로 계층 구조로 정렬됩니다.

역할	설명
소유자	Admin 역할의 모든 권한이 있으며 Astra 계정도 삭제할 수 있습니다.
관리자	에는 구성원 역할의 모든 권한이 있으며, 사용자를 초대하여 계정에 참가할 수도 있습니다.
회원	Astra 애플리케이션 및 컴퓨팅 리소스를 완벽하게 관리할 수 있습니다.
Viewer(뷰어)	리소스를 볼 수만 있습니다.

네임스페이스 세분화를 통해 RBAC 강화

이 기능은 Astra REST API 22.04 릴리스에 도입되었습니다.

특정 사용자에 대해 역할 바인딩을 설정할 때 제약 조건을 적용하여 사용자가 액세스할 수 있는 네임스페이스를 제한할 수 있습니다. 이 제약 조건은 아래 표에 설명된 대로 여러 가지 방법으로 정의할 수 있습니다. 자세한 내용은 역할 바인딩 API에서 roleContraints 매개 변수를 참조하십시오.

네임스페이스	설명
모두	사용자는 와일드카드 매개 변수 " * ""를 통해 모든 네임스페이스에 액세스할 수 있습니다. 이 값은 이전 버전과의 호환성을 유지하기 위한 기본값입니다.
없음	제약 조건 목록은 비어 있지만 지정됩니다. 이는 사용자가 네임스페이스에 액세스할 수 없음을 나타냅니다.
네임스페이스 목록	네임스페이스 UUID가 포함되어 있어 사용자를 단일 네임스페이스로 제한합니다. 쉼표로 구분된 목록을 사용하여 여러 네임스페이스에 액세스할 수도 있습니다.
라벨	레이블이 지정되고 일치하는 모든 네임스페이스에 대한 액세스가 허용됩니다.

컬렉션 작업

Astra Control REST API는 정의된 쿼리 매개 변수를 통해 리소스 컬렉션에 액세스하는 여러 가지 방법을 제공합니다.

값 선택

Include 매개 변수를 사용하여 각 리소스 인스턴스에 대해 반환할 키 값 쌍을 지정할 수 있습니다. 모든 인스턴스가 응답 바디에 반환됩니다.

필터링

컬렉션 리소스 필터링을 사용하면 API 사용자가 리소스가 응답 본문에 반환되는지 여부를 결정하는 조건을 지정할 수 있습니다. 필터 파라미터는 필터링 조건을 나타내는 데 사용된다.

정렬

컬렉션 리소스 정렬을 사용하면 API 사용자가 응답 본문에서 리소스가 반환되는 순서를 지정할 수 있습니다. OrderBy 매개변수는 필터링 조건을 나타내는 데 사용됩니다.

페이지 매김

"limit" 매개 변수를 사용하여 요청에 대해 반환되는 리소스 인스턴스 수를 제한하여 페이지 매김 기능을 적용할 수 있습니다.

카운트

TRUE로 설정된 부울 매개 변수 'count'를 포함하면 지정된 응답에 대해 반환된 배열의 리소스 수가 메타데이터 섹션에 제공됩니다.

진단 및 지원

Astra Control REST API에서 진단 및 디버깅에 사용할 수 있는 몇 가지 지원 기능이 있습니다.

API 리소스

API 리소스를 통해 진단 정보와 지원을 제공하는 몇 가지 Astra 기능이 제공됩니다.

유형	설명
이벤트	Astra 처리의 일부로 기록된 시스템 활동.
통지	사용자에게 제공할 만큼 중요한 것으로 간주되는 이벤트의 하위 집합입니다.
읽지 않은 알림	사용자가 읽거나 검색할 수 없는 알림입니다.

API 토큰을 취소합니다

더 이상 필요하지 않은 경우 Astra 웹 인터페이스에서 API 토큰을 취소할 수 있습니다.

시작하기 전에

Astra 계정이 필요합니다. 취소할 토큰도 식별해야 합니다.

이 작업에 대해

토큰이 취소되면 즉시 영구적으로 사용할 수 없게 됩니다.

단계

1. 계정 자격 증명을 사용하여 Astra에 로그인합니다.

   Astra Control Service에 대한 다음 사이트에 액세스합니다. "https://astra.netapp.io"

2. 페이지 오른쪽 상단의 그림 아이콘을 클릭하고 * API access * 를 선택합니다.

3. 취소할 토큰을 선택합니다.
4. 작업 * 드롭다운 상자에서 * 토큰 해지 * 를 클릭합니다.

인프라 워크플로우

시작하기 전에

이러한 워크플로우를 사용하여 Astra Control Center 배포 모델에 사용되는 인프라를 생성하고 유지 관리할 수 있습니다. 대부분의 경우 워크플로도 Astra Control Service와 함께 사용할 수 있습니다.

> 이러한 워크플로우는 NetApp이 언제든지 확장 및 개선할 수 있으므로 정기적으로 검토해야 합니다.

일반 준비

Astra 워크플로를 사용하기 전에 반드시 검토하십시오 "워크플로우 사용을 준비하십시오".

워크플로 범주

인프라 워크플로우는 다양한 범주로 구성되어 있어 원하는 워크플로를 쉽게 찾을 수 있습니다.

범주	설명
ID 및 액세스	이러한 워크플로를 통해 ID 및 Astra 액세스 방법을 관리할 수 있습니다. 리소스에는 사용자, 자격 증명 및 토큰이 포함됩니다.
버킷	이러한 워크플로우를 사용하여 백업을 저장하는 데 사용되는 S3 버킷을 생성하고 관리할 수 있습니다.
스토리지	이러한 워크플로우를 사용하여 스토리지 백엔드 및 볼륨을 추가하고 유지할 수 있습니다.
클러스터	관리 Kubernetes 클러스터를 추가하여 포함된 애플리케이션을 보호하고 지원할 수 있습니다.

ID 및 액세스

사용자를 나열합니다

특정 Astra 계정에 대해 정의된 사용자를 나열할 수 있습니다.

사용자를 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/core/v1/users

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
포함	쿼리	아니요	필요에 따라 응답에서 반환할 값을 선택합니다.

curl 예: 모든 사용자의 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

curl 예: 모든 사용자의 이름, 성 및 ID를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users?include=first
Name,lastName,id' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

JSON 출력 예

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

버킷

버킷 나열

특정 Astra 계정에 대해 정의된 S3 버킷을 나열할 수 있습니다.

버킷을 나열하십시오

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/topology/v1/버킷

CURL 예: 모든 버킷에 대한 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/buckets'
--header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

스토리지

저장소 백엔드를 나열합니다

사용 가능한 저장소 백엔드를 나열할 수 있습니다.

버킷을 나열하십시오

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/topology/v1/storageBackends

curl 예: 모든 저장소 백엔드에 대한 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/storageBackends
' --header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    {
      "backendCredentialsName": "10.191.77.177",
      "backendName": "myinchunhcluster-1",
      "backendType": "ONTAP",
      "backendVersion": "9.8.0",
      "configVersion": "Not applicable",
      "health": "Not applicable",
      "id": "46467c16-1585-4b71-8e7f-f0bc5ff9da15",
      "location": "nalab2",
      "metadata": {
        "createdBy": "4c483a7e-207b-4f9a-87b7-799a4629d7c8",
        "creationTimestamp": "2021-07-30T14:26:19Z",
        "modificationTimestamp": "2021-07-30T14:26:19Z"
      },
      "ontap": {
        "backendManagementIP": "10.191.77.177",
        "managementIPs": [
          "10.191.77.177",
          "10.191.77.179"
        ]
      },
      "protectionPolicy": "Not applicable",
      "region": "Not applicable",
      "state": "Running",
      "stateUnready": [],
      "type": "application/astra-storageBackend",
      "version": "1.0",
      "zone": "Not applicable"
    }
  ]
}
```


클러스터

관리되는 클러스터를 나열합니다

현재 Astra에서 관리하는 Kubernetes 클러스터를 나열할 수 있습니다.

클러스터를 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/topology/v1/managedClusters

curl 예: 모든 클러스터의 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/managedClusters
' --header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

관리 워크플로

시작하기 전에

Astra 관리 클러스터 내에서 애플리케이션을 관리하는 과정에서 이러한 워크플로우를 사용할 수 있습니다.

> 이러한 워크플로우는 NetApp이 언제든지 확장 및 개선할 수 있으므로 정기적으로 검토해야 합니다.

일반 준비

Astra 워크플로를 사용하기 전에 반드시 검토하십시오 "워크플로우 사용을 준비하십시오".

워크플로 범주

관리 워크플로는 원하는 항목을 쉽게 찾을 수 있도록 여러 범주로 구성됩니다.

범주	설명
애플리케이션 제어	이러한 워크플로를 통해 관리되는 응용 프로그램과 관리되지 않는 응용 프로그램을 제어할 수 있습니다. 앱을 나열하고 관리되는 앱을 만들고 제거할 수 있습니다.
애플리케이션 보호	이러한 워크플로우를 사용하여 스냅샷과 백업을 통해 관리되는 애플리케이션을 보호할 수 있습니다.
앱 클론 생성 및 복원	이러한 워크플로에서는 관리되는 애플리케이션을 클론 복제 및 복원하는 방법을 설명합니다.
지원	일반 Kubernetes 환경뿐만 아니라 애플리케이션을 디버깅 및 지원하는 데 사용할 수 있는 몇 가지 워크플로우가 있습니다.

추가 고려 사항

관리 워크플로를 사용할 때는 몇 가지 추가 고려 사항이 있습니다.

앱 복제

애플리케이션 클론 생성 시 고려해야 할 몇 가지 사항은 다음과 같습니다. 아래 설명된 매개 변수는 JSON 입력의 일부입니다.

소스 클러스터 식별자입니다

'소스 클러스터 ID' 값은 항상 원래 앱이 설치된 클러스터를 식별합니다.

클러스터 식별자입니다

'clusterID' 값은 새 앱을 설치할 클러스터를 식별합니다.

- 같은 클러스터 내에서 복제하면 clusterID와 sourceClusterID가 같은 값을 갖습니다.
- 클러스터 간에 클론을 생성할 때 두 값은 서로 다르며 클러스터 ID는 타겟 클러스터의 ID가 되어야 합니다.

네임스페이스

네임스페이스 값은 원본 소스 앱과 달라야 합니다. 또한 클론의 네임스페이스가 존재할 수 없으며 Astra에서 생성합니다.

백업 및 스냅샷

"backupID" 또는 "shapshotID" 매개 변수를 사용하여 기존 백업 또는 스냅샷에서 응용 프로그램을 복제할 수도 있습니다. 백업 또는 스냅샷을 제공하지 않는 경우 먼저 애플리케이션의 백업을 생성한 다음 백업에서 클론을 생성합니다.

앱을 복원하는 중입니다

다음은 응용 프로그램을 복원할 때 고려해야 할 몇 가지 사항입니다.

- 애플리케이션 복구는 클론 작업과 매우 유사합니다.
- 앱을 복구할 때는 백업 또는 스냅샷을 제공해야 합니다.

애플리케이션 제어

관리되지 않는 앱을 나열합니다

현재 Astra에서 관리하지 않는 애플리케이션을 나열할 수 있습니다. 이 작업은 관리할 앱을 선택하는 과정에서 수행할 수 있습니다.

> 이러한 워크플로에 사용되는 REST 끝점은 기본적으로 모든 Astra 애플리케이션을 반환합니다. API 호출에서 'filter' query 매개 변수를 사용하여 관리되지 않는 앱만 반환하도록 요청할 수 있습니다. 또는 필터 매개 변수를 생략하여 모든 앱을 반환한 다음 출력에서 'managedState' 필드를 검사하여 '관리되지 않는' 상태에 있는 앱을 확인할 수도 있습니다.

관리되는 상태가 관리되지 않는 것과 동일한 앱만 나열합니다

이 워크플로에서는 필터 쿼리 매개 변수를 사용하여 관리되지 않는 앱만 반환합니다.

관리되지 않는 응용 프로그램을 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/topology/v1/apps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
필터	쿼리	아니요	필터를 사용하여 반환할 앱을 지정합니다.
포함	쿼리	아니요	필요에 따라 응답에서 반환할 값을 선택합니다.

curl 예제: 관리되지 않는 앱에 대한 이름, id 및 managedState를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/apps?filter=man
agedState%20eq%20'unmanaged'&include=name,id,managedState' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    [
      "maria",
      "eed19f78-0884-4792-bb7a-313258c6b0b1",
      "unmanaged"
    ],
    [
      "test-postgres-app",
      "1ee6235b-cda1-45cb-8d4c-630bdb8b41a5",
      "unmanaged"
    ],
    [
      "postgres1-postgresql",
      "e591ee59-ea90-4a9f-8e6c-d2b6e8647096",
      "unmanaged"
    ],
    [
      "kube-system",
      "077a2f73-4b51-4d04-8c6c-f63b3b069755",
      "unmanaged"
    ],
    [
      "trident",
      "5b6fc28f-e308-4653-b9d2-6d66a764d2e1",
      "unmanaged"
    ],
    [
      "postgres1-postgresql-clone",
      "06be05c5-763e-4d73-bd06-1f27f5f2e130",
      "unmanaged"
    ]
  ],
  "metadata": {}
}
```


모든 앱을 나열하고 관리되지 않는 앱을 선택합니다

이 워크플로는 모든 앱을 반환합니다. 관리되지 않는 출력을 확인하려면 출력을 검사해야 합니다.

모든 응용 프로그램을 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/topology/v1/apps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
포함	쿼리	아니요	필요에 따라 응답에서 반환할 값을 선택합니다.

curl 예: 모든 앱에 대한 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/apps' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

curl 예제: 모든 앱에 대해 **name , id** 및 **managedState** 를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/topology/v1/apps?include=na
me,id,managedState' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    [
      "maria",
      "eed19f78-0884-4792-bb7a-313258c6b0b1",
      "unmanaged"
    ],
    [
      "mariadb-mariadb",
      "8da20fff-c69c-4170-bb0d-e4f91c5a1333",
      "managed"
    ],
    [
      "test-postgres-app",
      "1ee6235b-cda1-45cb-8d4c-630bdb8b41a5",
      "unmanaged"
    ],
    [
      "postgres1-postgresql",
      "e591ee59-ea90-4a9f-8e6c-d2b6e8647096",
      "unmanaged"
    ],
    [
      "kube-system",
      "077a2f73-4b51-4d04-8c6c-f63b3b069755",
      "unmanaged"
    ],
    [
      "trident",
      "5b6fc28f-e308-4653-b9d2-6d66a764d2e1",
      "unmanaged"
    ],
    [
      "postgres1-postgresql-clone",
      "06be05c5-763e-4d73-bd06-1f27f5f2e130",
      "unmanaged"
    ],
    [
      "davidns-postgres-app",
      "11e046b7-ec64-4184-85b3-debcc3b1da4d",
      "managed"
    ]
  ],
  "metadata": {}
}
```

관리되지 않는 응용 프로그램을 선택합니다

API 호출 출력을 검토하고 'unmanaged'와 같은 'managedState'가 있는 앱을 수동으로 선택합니다.

관리되는 앱을 나열합니다

현재 Astra에서 관리하는 애플리케이션을 나열할 수 있습니다. 특정 앱의 스냅샷 또는 백업을 찾는 과정에서 이 작업을 수행할 수 있습니다.

응용 프로그램을 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/k8s/v1/managedApps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
포함	쿼리	아니요	필요에 따라 응답에서 반환할 값을 선택합니다.

curl 예: 모든 앱에 대한 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps'
--header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

curl 예: 모든 앱의 이름, **ID** 및 상태를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps?include=
name,id,state' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    [
      "test-postgres-app",
      "1ee6235b-cda1-45cb-8d4c-630bdb8b41a5",
      "running"
    ]
  ],
  "metadata": {}
}
```

관리되는 앱을 가져옵니다

관리되는 단일 응용 프로그램을 설명하는 모든 리소스 변수를 검색할 수 있습니다.

시작하기 전에

검색할 관리 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

1.응용 프로그램을 가져옵니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	검색할 관리되는 응용 프로그램의 ID 값입니다.

curl 예: 응용 프로그램에 대한 모든 데이터를 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

앱을 관리합니다

이미 Astra로 알려진 애플리케이션을 기반으로 관리되는 애플리케이션을 만들 수 있습니다. 애플리케이션이 관리되면 일반 백업 및 스냅샷을 생성하여 보호할 수 있습니다.

시작하기 전에

관리할 검색된 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되지 않는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

애플리케이션을 관리합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/account/{AccountID}/k8s/v1/managedApps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
JSON을 참조하십시오	바디	예	관리할 응용 프로그램을 식별하는 데 필요한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-managedApp",
  "version": "1.1",
  "id": "7da20fff-c69d-4270-bb0d-a4f91c5a1333"
}
```

CURL 예: 앱 관리

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps'
--header 'Content-Type: application/astra-managedApp+json' --header
'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'  --d @JSONinput
```

앱 관리를 취소합니다

더 이상 필요하지 않은 경우 관리되는 앱을 제거할 수 있습니다. 관리되는 응용 프로그램을 제거하면 연결된 스케줄도 삭제됩니다.

시작하기 전에

관리를 취소할 관리 앱 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

애플리케이션의 백업 및 스냅샷은 삭제되어도 자동으로 제거되지 않습니다. 더 이상 백업 및 스냅샷이 필요하지 않은 경우, 응용 프로그램을 제거하기 전에 삭제해야 합니다.

1.앱을 비관리했습니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
삭제	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	제거할 관리되는 응용 프로그램을 식별합니다.

curl 예**:** 관리되는 앱을 제거합니다

```
curl --location -i --request DELETE
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

애플리케이션 보호

스냅샷을 나열합니다

특정 관리되는 애플리케이션에 대해 생성된 스냅샷을 나열할 수 있습니다.

시작하기 전에

스냅샷을 나열할 관리되는 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appSnaps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	나열된 스냅샷을 소유하는 관리되는 애플리케이션을 식별합니다.
카운트	쿼리	아니요	count=true이면 응답의 메타데이터 섹션에 스냅샷 수가 포함됩니다.

curl 예: 앱의 모든 스냅샷을 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appSnaps' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>'
```

curl 예: 앱과 카운트에 대한 모든 스냅샷을 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appSnaps?count=true' --header 'Accept: */*' --header
'Authorization: Bearer <API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    {
      "id": "dc2974ae-f71d-4c81-91b5-f96cf72dc3ba",
      "metadata": {
        "createdBy": "fb093413-b6fc-4a64-a48a-afc32ada8537",
        "creationTimestamp": "2021-06-04T21:23:14Z",
        "modificationTimestamp": "2021-06-04T21:23:14Z",
        "labels": []
      },
      "snapshotAppAsset": "4547658d-cc06-4c1d-ad8a-4a05274d0db0",
      "snapshotCreationTimestamp": "2021-06-04T21:23:47Z",
      "name": "test-postgres-app-snapshot-20210604212213",
      "state": "completed",
      "stateUnready": [],
      "type": "application/astra-appSnap",
      "version": "1.0"
    }
  ],
  "metadata": {
    "count": 1
  }
}
```


백업을 나열합니다

특정 관리되는 애플리케이션에 대해 생성된 백업을 나열할 수 있습니다.

시작하기 전에

백업을 나열할 관리되는 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

백업을 나열합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appBackups

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	나열된 백업을 소유하는 관리되는 애플리케이션을 식별합니다.

curl 예: 앱의 모든 백업을 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appBackups' --header 'Accept: */*' --header 'Authorization:
Bearer <API_TOKEN>'
```

JSON 출력 예


```
{
  "items": [
    {
      "type": "application/astra-appBackup",
      "version": "1.0",
      "id": "ed39fdb0-12db-497b-9e46-20036c1fb0d2",
      "name": "mariadb-mariadb-backup-20210617175900",
      "state": "completed",
      "stateUnready": [],
      "bytesDone": 0,
      "percentDone": 100,
      "metadata": {
        "labels": [],
        "creationTimestamp": "2021-06-17T17:59:09Z",
        "modificationTimestamp": "2021-06-17T17:59:09Z",
        "createdBy": "fb093413-b6fc-4a64-a48a-afc32ada8537"
      }
    }
  ],
  "metadata": {}
}
```


관리되는 앱에 대한 스냅샷을 생성합니다

관리되는 특정 응용 프로그램에 대한 스냅샷을 생성할 수 있습니다.

시작하기 전에

스냅샷을 생성하려는 관리되는 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

스냅샷을 생성합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appSnaps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	스냅샷이 생성될 관리되는 애플리케이션을 식별합니다.
JSON을 참조하십시오	바디	예	스냅샷에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-appSnap",
  "version": "1.0",
  "name": "snapshot-david-1"
}
```

curl 예: 앱에 대한 스냅샷을 생성합니다

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appSnaps' --header 'Content-Type: application/astra-appSnap+json'
--header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>' --d
@JSONinput
```

관리되는 앱에 대한 백업을 생성합니다

특정 관리되는 응용 프로그램에 대한 백업을 만들 수 있습니다. 백업을 사용하여 앱을 복원 또는 클론 복제할 수 있습니다.

시작하기 전에

백업을 만들려는 관리되는 앱의 ID가 있어야 합니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.

백업을 생성합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appBackups

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	백업을 생성할 관리되는 애플리케이션을 식별합니다.
JSON을 참조하십시오	바디	예	백업에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-appBackup",
  "version": "1.0",
  "name": "backup-david-1"
}
```

curl 예: 앱의 백업을 만듭니다

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appBackups' --header 'Content-Type: application/astra-
appBackup+json' --header 'Accept: */*' --header 'Authorization: Bearer
<API_TOKEN>' --d @JSONinput
```

스냅샷을 삭제합니다

관리되는 응용 프로그램과 연결된 스냅샷을 삭제할 수 있습니다.

시작하기 전에

다음 항목이 있어야 합니다.

- 스냅샷을 소유하는 관리되는 앱의 ID입니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.
- 삭제할 스냅샷의 ID입니다. 필요한 경우 워크플로를 사용할 수 있습니다 "스냅샷을 나열합니다" 스냅샷을 찾습니다.

스냅샷을 삭제합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
삭제	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appSnaps/{appSnap_id}

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	스냅샷을 소유하는 관리되는 애플리케이션을 식별합니다.
스냅샷 ID입니다	경로	예	삭제할 스냅샷을 식별합니다.

curl 예: 앱에 대한 스냅샷 하나를 삭제합니다

```
curl --location -i --request DELETE
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appSnaps/<SNAPSHOT_ID>' --header 'Accept: */*' --header
'Authorization: Bearer <API_TOKEN>'
```

백업을 삭제합니다

관리되는 응용 프로그램과 연결된 백업을 삭제할 수 있습니다.

시작하기 전에

다음 항목이 있어야 합니다.

- 백업을 소유하는 관리되는 앱의 ID입니다. 필요한 경우 워크플로를 사용할 수 있습니다 "관리되는 앱을 나열합니다" 를 눌러 응용 프로그램을 찾습니다.
- 삭제할 백업의 ID입니다. 필요한 경우 워크플로를 사용할 수 있습니다 "백업을 나열합니다" 스냅샷을 찾습니다.

백업을 삭제합니다

다음과 같은 REST API 호출을 수행한다.

아래 설명된 대로 선택적 요청 헤더를 사용하여 실패한 백업을 강제로 삭제할 수 있습니다.

HTTP 메소드	경로
삭제	/accounts/{account_id}/k8s/v1/managedApps/{managedApp_id}/appBackups/{appBackup_id}

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
관리되는 앱 ID입니다	경로	예	백업을 소유하는 관리되는 애플리케이션을 식별합니다.
백업 ID입니다	경로	예	삭제할 백업을 식별합니다.
강제 삭제	머리글	아니요	실패한 백업을 강제로 삭제하는 데 사용됩니다.

curl 예: 앱에 대한 단일 백업을 삭제합니다

```
curl --location -i --request DELETE
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appBackups/<BACKUP_ID>' --header 'Accept: */*' --header
'Authorization: Bearer <API_TOKEN>'
```

curl 예: **force** 옵션을 사용하여 앱에 대한 단일 백업을 삭제합니다

```
curl --location -i --request DELETE
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<MANAGED
_APP_ID>/appBackups/<BACKUP_ID>' --header 'Accept: */*' --header
'Authorization: Bearer <API_TOKEN>' --header 'Force-Delete: true'
```

앱 복제 및 복원

관리되는 앱을 복제합니다

기존 관리 앱을 복제하여 새 애플리케이션을 만들 수 있습니다.

시작하기 전에

이 워크플로에 대해 다음 사항을 참고하십시오.

- 앱 백업 또는 스냅샷이 사용되지 않습니다
- 클론 작업은 동일한 클러스터 내에서 수행됩니다

앱을 다른 클러스터로 클론 복제하려면 환경에 맞게 JSON 입력의 'clusterId' 매개 변수를 업데이트해야 합니다.

복제할 관리 대상 앱을 선택합니다

워크플로우를 수행합니다 "관리되는 앱을 나열합니다" 을 클릭하고 복제할 애플리케이션을 선택합니다. 앱을 복제하는 데 사용되는 나머지 통화에는 몇 가지 리소스 값이 필요합니다.

앱을 복제합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/account/{AccountID}/k8s/v1/managedApps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
JSON을 참조하십시오	바디	예	복제된 앱에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-managedApp",
  "version": "1.0",
  "name": "postgres1-postgresql-clone",
  "clusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "sourceClusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "namespace": "davidns-postgres-app",
  "sourceAppID": "e591ee59-ea90-4a9f-8e6c-d2b6e8647096"
}
```

컬링 예: 앱 복제

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps'
--header 'Content-Type: application/astra-managedApp+json' --header '*/*'
--header 'Authorization: Bearer <API_TOKEN>' --d @JSONinput
```

스냅숏에서 관리되는 응용 프로그램의 클론을 생성합니다

앱 스냅샷에서 새 애플리케이션을 복제하여 생성할 수 있습니다.

시작하기 전에

이 워크플로에 대해 다음 사항을 참고하십시오.

- 앱 스냅샷이 사용됩니다
- 클론 작업은 동일한 클러스터 내에서 수행됩니다

> 앱을 다른 클러스터로 클론 복제하려면 환경에 맞게 JSON 입력의 'clusterId' 매개 변수를 업데이트해야 합니다.

복제할 관리 대상 앱을 선택합니다

워크플로우를 수행합니다 "관리되는 앱을 나열합니다" 을 클릭하고 복제할 애플리케이션을 선택합니다. 앱을 복제하는 데 사용되는 나머지 통화에는 몇 가지 리소스 값이 필요합니다.

사용할 스냅샷을 선택합니다

워크플로우를 수행합니다 "스냅샷을 나열합니다" 을 클릭하고 사용할 스냅샷을 선택합니다.

앱을 복제합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/account/{AccountID}/k8s/v1/managedApps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
JSON을 참조하십시오	바디	예	복제된 앱에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-managedApp",
  "version": "1.0",
  "name": "postgres1-postgresql-clone",
  "clusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "sourceClusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "namespace": "davidns-postgres-app",
  "snapshotID": "e24515bd-a28e-4b28-b832-f3c74dbf32fb",
  "sourceAppID": "e591ee59-ea90-4a9f-8e6c-d2b6e8647096"
}
```

curl 예: 스냅샷에서 앱을 복제합니다

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps'
--header 'Content-Type: application/astra-managedApp+json' --header '*/*'
--header 'Authorization: Bearer <API_TOKEN>' --d @JSONinput
```

백업에서 관리되는 앱을 복제합니다

앱 백업에서 새 관리 애플리케이션을 복제하여 생성할 수 있습니다.

시작하기 전에

이 워크플로에 대해 다음 사항을 참고하십시오.

- 앱 백업이 사용됩니다
- 클론 작업은 동일한 클러스터 내에서 수행됩니다

> 앱을 다른 클러스터로 클론 복제하려면 환경에 맞게 JSON 입력의 'clusterId' 매개 변수를 업데이트해야 합니다.

복제할 관리 대상 앱을 선택합니다

워크플로우를 수행합니다 "관리되는 앱을 나열합니다" 을 클릭하고 복제할 애플리케이션을 선택합니다. 앱을 복제하는 데 사용되는 나머지 통화에는 몇 가지 리소스 값이 필요합니다.

사용할 백업을 선택합니다

워크플로우를 수행합니다 "백업을 나열합니다" 을 클릭하고 사용할 백업을 선택합니다.

앱을 복제합니다

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
게시	/account/{AccountID}/k8s/v1/managedApps

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
JSON을 참조하십시오	바디	예	복제된 앱에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-managedApp",
  "version": "1.0",
  "name": "postgres1-postgresql-clone",
  "clusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "sourceClusterID": "30880586-d579-4d27-930f-a9633e59173b",
  "namespace": "davidns-postgres-app",
  "backupID": "e24515bd-a28e-4b28-b832-f3c74dbf32fb",
  "sourceAppID": "e591ee59-ea90-4a9f-8e6c-d2b6e8647096"
}
```

curl 예: 백업에서 앱을 복제합니다

```
curl --location -i --request POST
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps'
--header 'Content-Type: application/astra-managedApp+json' --header '*/*'
--header 'Authorization: Bearer <API_TOKEN>' --d @JSONinput
```

백업에서 관리되는 앱을 복원합니다

백업에서 새 앱을 만들어 관리되는 응용 프로그램을 복원할 수 있습니다.

복구할 관리되는 앱을 선택합니다

워크플로우를 수행합니다 "관리되는 앱을 나열합니다" 을 클릭하고 복제할 애플리케이션을 선택합니다. 앱을 복제하는 데 사용되는 나머지 통화에는 몇 가지 리소스 값이 필요합니다.

사용할 백업을 선택합니다

워크플로우를 수행합니다 "백업을 나열합니다" 을 클릭하고 사용할 백업을 선택합니다.

앱을 복원합니다

다음과 같은 REST API 호출을 수행한다. 백업(아래 그림 참조) 또는 스냅샷의 ID를 제공해야 합니다.

HTTP 메소드	경로
를 누릅니다	/account/{AccountID}/k8s/v1/managedApps/{appID}

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
JSON을 참조하십시오	바디	예	복제된 앱에 대한 매개 변수를 제공합니다. 아래 예를 참조하십시오.

JSON 입력 예

```
{
  "type": "application/astra-managedApp",
  "version": "1.2",
  "backupID": "e24515bd-a28e-4b28-b832-f3c74dbf32fb"
}
```

CURL 예: 백업에서 앱을 복원하십시오

```
curl --location -i --request PUT
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/k8s/v1/managedApps/<APP_ID>
' --header 'Content-Type: application/astra-managedApp+json' --header
'*/*' --header 'ForceUpdate: true' --header 'Authorization: Bearer
<API_TOKEN>' --d @JSONinput
```

지원

알림을 나열합니다

특정 Astra 계정에 대한 알림을 나열할 수 있습니다. 시스템 작업을 모니터링하거나 문제를 디버깅하는 과정에서 이 작업을 수행할 수 있습니다.

다음과 같은 REST API 호출을 수행한다.

HTTP 메소드	경로
가져오기	/account/{AccountID}/core/v1/notifications

추가 입력 매개변수

모든 REST API 호출에서 일반적으로 사용되는 매개 변수 외에도 이 단계의 curl 예제에도 다음 매개 변수가 사용됩니다.

매개 변수	유형	필수 요소입니다	설명
필터	쿼리	아니요	필요에 따라 응답에서 반환하려는 알림을 필터링합니다.
포함	쿼리	아니요	필요에 따라 응답에서 반환할 값을 선택합니다.

curl 예**:** 모든 알림을 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/notifications'
--header 'Accept: */*' --header 'Authorization: Bearer <API_TOKEN>'
```

curl 예**:** 경고의 심각도를 나타내는 알림에 대한 설명을 반환합니다

```
curl --location -i --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/notifications?filte
r=severity%20eq%20'warning'&include=description' --header 'Accept: */*'
--header 'Authorization: Bearer <API_TOKEN>'
```

JSON 출력 예

```
{
  "items": [
    [
      "Trident on cluster david-ie-00 has failed or timed out;
installation of the Trident operator failed or is not yet complete;
operator failed to reach an installed state within 300.00 seconds;
container trident-operator not found in operator deployment"
    ],
    [
      "Trident on cluster david-ie-00 has failed or timed out;
installation of the Trident operator failed or is not yet complete;
operator failed to reach an installed state within 300.00 seconds;
container trident-operator not found in operator deployment"
    ]
  ],
  "metadata": {}
}
```


실패한 앱을 삭제합니다

실패한 상태의 백업 또는 스냅샷이 있는 관리되는 앱을 제거할 수 없습니다. 이 경우 아래 설명된 워크플로를 사용하여 앱을 수동으로 제거할 수 있습니다.

삭제할 관리되는 앱을 선택합니다

워크플로우를 수행합니다 "관리되는 앱을 나열합니다" 제거할 응용 프로그램을 선택합니다.

앱의 기존 백업을 나열합니다

워크플로우를 수행합니다 "백업을 나열합니다".

모든 백업을 삭제합니다

워크플로를 수행하여 모든 앱 백업을 삭제합니다 "백업을 삭제합니다" 를 선택합니다.

앱의 기존 스냅샷을 나열합니다

워크플로우를 수행합니다 "스냅샷을 나열합니다".

모든 스냅샷을 삭제합니다

워크플로우를 수행합니다 "스냅샷을 삭제합니다" 를 선택합니다.

응용 프로그램을 제거합니다

워크플로우를 수행합니다 "앱 관리를 취소합니다" 를 눌러 응용 프로그램을 제거합니다.

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, 하나의 "astraSDK.py" 파일로 제공됩니다. 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 파일 'astraSDK.py'의 핵심 Python 클래스를 나타냅니다.

툴킷 스크립트입니다

Astra SDK 파일 외에 toolkit.py 스크립트도 사용할 수 있다. 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" 이름아래 * NetApp-Astra-툴킷 **.** 패키지에 버전 번호가 할당되며 필요에 따라 계속 업데이트됩니다**. PIP** 패키지 관리 유틸리티를 사용하여 사용자 환경에 패키지를 설치해야 합니다.

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 있는 Astra SDK)
- toolkit.py(상위 수준 함수 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.

- Pip을 사용하여 PyPI에서 Python 환경에 패키지를 설치합니다
- Git Hub 리포지토리를 클론하고 다음 중 하나를 수행합니다.
  - 패키지를 Docker 컨테이너(필요한 모든 것을 포함)로 구축
  - Python 클라이언트 코드에 액세스할 수 있도록 두 개의 핵심 Python 파일을 복사합니다

자세한 내용은 PyPI 및 GitHub 페이지를 참조하십시오.

Astra Control 환경에 대한 요구 사항

Astra SDK에서 Python 클래스를 직접 사용하거나 toolkit.py 스크립트의 함수를 사용하는 경우 Astra Control 구축 시 REST API에 액세스할 수 있습니다. 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 SDK의 요구사항

SDK에는 로컬 Python 환경과 관련된 몇 가지 사전 요구 사항이 있습니다. 예를 들어 Python 3.5 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"

기본 **Python**

시작하기 전에

Python은 특히 데이터 센터 자동화에 널리 사용되는 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, https://astra.netapp.io` 에 있는 서비스에 연결할 수 있어야 합니다.

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. '-i'(또는 '--identity') CLI 매개변수를 사용하여 이 파일을 Python 스크립트로 전달해야 합니다.

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

관리되는 앱을 나열합니다

다음 스크립트를 사용하여 Astra 계정에 대해 관리되는 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 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)
```

API 참조입니다

HTTP 메서드, 입력 매개 변수 및 응답을 포함하여 모든 Astra Control REST API 호출의 세부 정보에 액세스할 수 있습니다. 이 전체 참조는 REST API를 사용하여 자동화 응용 프로그램을 개발할 때 유용합니다.

REST API 참조 문서는 현재 Astra Control과 함께 제공되며 온라인으로 제공됩니다.

시작하기 전에

Astra Control Center 또는 Astra Control Service 계정이 필요합니다.

단계

1. 계정 자격 증명을 사용하여 Astra에 로그인합니다.

   Astra Control Service에 대한 다음 사이트에 액세스합니다. "https://astra.netapp.io"

2. 페이지 오른쪽 상단의 그림 아이콘을 클릭하고 * API access * 를 선택합니다.
3. 페이지 맨 위에서 * API Documentation * 아래에 표시된 URL을 클릭합니다.
4. 메시지가 나타나면 계정 자격 증명을 다시 입력합니다.

추가 리소스

일반 REST 및 클라우드 개념뿐 아니라 NetApp 클라우드 서비스 및 지원에 대한 자세한 정보를 얻고 도움을 받을 수 있는 추가 리소스가 있습니다.

아스트라

- "Astra Control Center 22.04 문서"

  고객 구내에 구축된 Astra Control Center 소프트웨어의 현재 릴리스에 대한 문서입니다.

- "Astra Control Service 문서"

  퍼블릭 클라우드에서 사용 가능한 Astra Control Service 소프트웨어의 현재 릴리스에 대한 문서입니다.

- "Astra Trident 문서"

  NetApp에서 관리하는 오픈 소스 스토리지 오케스트레이터인 Astra Trident 소프트웨어의 현재 릴리즈에 대한 문서입니다.

- "Astra 제품군 문서"

  온프레미스 및 퍼블릭 클라우드 구축을 위한 모든 Astra 설명서를 액세스할 수 있는 중앙 위치입니다.

NetApp 클라우드 리소스

- "NetApp 클라우드 솔루션"

  NetApp 클라우드 솔루션을 위한 중앙 사이트

- "NetApp Cloud Central 콘솔"

  NetApp Cloud Central 서비스 콘솔(로그인 포함)

- "NetApp 지원"

  문제 해결 도구, 문서 및 기술 지원 지원을 이용할 수 있습니다.

REST 및 클라우드 개념

- 박사 "학위 논문" Roy Fielding

  이 책은 REST 응용 프로그램 개발 모델을 소개하고 설정했습니다.

- "Auth0"

  웹 액세스를 위해 Astra 서비스에서 사용하는 인증 및 권한 부여 플랫폼 서비스입니다.

- "RFC 편집기"

  고유 번호가 지정된 RFC 문서의 모음으로 유지 관리되는 웹 및 인터넷 표준에 대한 신뢰할 수 있는 소스입니다.

이전 버전의 **Astra Control Automation** 설명서

아래 링크를 통해 이전 Astra Control 릴리즈에 대한 자동화 문서에 액세스할 수 있습니다.

- "Astra Control Automation 21.12 문서"
- "Astra Control Automation 21.08 문서"

법적 고지

법적 고지 사항은 저작권 선언, 상표, 특허 등에 대한 액세스를 제공합니다.

저작권

"https://www.netapp.com/company/legal/copyright/"

상표

NetApp, NetApp 로고, NetApp 상표 페이지에 나열된 마크는 NetApp Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.

"https://www.netapp.com/company/legal/trademarks/"

특허

NetApp 소유 특허 목록은 다음 사이트에서 확인할 수 있습니다.

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

개인 정보 보호 정책

"https://www.netapp.com/company/legal/privacy-policy/"

Astra Control API 라이센스

https://docs.netapp.com/us-en/astra-automation/media/astra-api-license.pdf