Astra Control Automation 22.04文件

Astra Automation 22.04

NetApp
December 04, 2023

目錄

Astra Control Automation 22.04文件

版本資訊

關於本版本

本網站的文件說明Astra Control的2022年4月（22.04）版Astra Control隨附的Astra Control REST API及相關自動化技術。特別是、此版本的REST API隨附於Astra Control Center和Astra Control Service的相應22.04版本。

請參閱下列頁面和網站、以取得本版本及先前版本的詳細資訊：

- "Astra Control REST API的新功能"
- "閒置資源和端點"
- "Astra Control Center 22.04文件"
- "Astra Control Service文件"
- "舊版Astra Automation文件"

Astra Control REST API的新功能

NetApp會定期更新Astra Control REST API、為您帶來新功能、增強功能和錯誤修復。

2022年4月26日（22.04）

此版本包括REST API的擴充與更新、以及增強的安全性與管理功能。

全新強化的**Astra**資源

新增兩種資源類型：套件*和*升級。此外、多個現有資源的版本也已升級。

增強的**RBAC**與命名空間精細度

將角色繫結至關聯的使用者時、您可以限制使用者可存取的命名空間。請參閱*角色繫結API*參考與 "RBAC安全性" 以取得更多資訊。

移除鏟斗

您可以在不再需要或無法正常運作的情況下移除貯體。

支援**Cloud Volumes ONTAP** 功能

現在支援將其作為儲存後端。Cloud Volumes ONTAP

其他產品增強功能

兩項Astra Control產品實作有多項額外增強功能、包括：

- Astra Control Center的一般入口

- 使用的私有叢集
- 支援Kubernetes 1.22
- 支援VMware Tanzu產品組合

請參閱Astra Control Center和Astra Control Service文件網站上的*新增功能*頁面。

相關資訊

- "Astra Control Center：最新消息"
- "Astra Control Service：最新消息"

2021年12月14日（21.12）

此版本包括擴充REST API、以及變更文件架構、以便透過未來的版本更新、更好地支援Astra Control的演進。

每個Astra Control版本都有獨立的Astra Automation文件

Astra Control的每個版本都包含一個獨特的REST API、經過強化並針對特定版本的功能量身打造。Astra Control REST API每個版本的文件現在都可在專屬網站及相關的GitHub內容儲存庫中取得。主文件網站 "Astra Control Automation" 永遠包含最新版本的文件。請參閱 "舊版Astra Control Automation文件" 以取得先前版本的相關資訊。

擴充REST資源類型

REST資源類型的數量持續增加、重點放在執行掛勾和儲存後端。新資源包括：帳戶、執行掛勾、掛機來源、執行掛勾置換、叢集節點、 託管儲存後端、命名空間、儲存設備和儲存節點。請參閱 "資源" 以取得更多資訊。

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK是開放原始碼套件、可讓您更輕鬆地為Astra Control環境開發自動化程式碼。核心是Astra SDK、其中包含一組類別、可抽象化REST API呼叫的複雜度。此外、還有一個工具組指令碼、可透過包裝和抽象化Python類別來執行特定的管理工作。請參閱 "NetApp Astra Control Python SDK" 以取得更多資訊。

2021年8月5日（21.08）

此版本包括引進新的Astra部署模式、以及REST API的重大擴充。

Astra Control Center部署模式

除了以公有雲端服務形式提供的現有Astra Control Service產品之外、此版本也包括Astra Control Center內部部署模式。您可以在站台上安裝Astra Control Center、以管理本機Kubernetes環境。這兩種Astra Control部署模式共用相同的REST API、但文件中所指出的細微差異較小。

擴充REST資源類型

透過Astra Control REST API存取的資源數量已大幅增加、許多新資源為內部部署的Astra Control Center產品提供了基礎。新資源包括：ASUP、權利、功能、授權、設定、 訂購、儲存庫、雲端、叢集、託管叢集、 儲存後端與儲存類別。請參閱 "資源" 以取得更多資訊。

支援**Astra**部署的其他端點

除了擴充的REST資源之外、還有其他幾個新的API端點可供支援Astra Control部署。

OpenAPI支援

OpenAPI端點可讓您存取目前的OpenAPI Json文件及其他相關資源。

OpenMetrics支援

OpenMetrics端點可透過OpenMetrics資源存取帳戶指標。

2021年4月15日（21.04）

此版本包含下列新功能與增強功能。

介紹**REST API**

Astra Control REST API可搭配Astra Control Service產品使用。這是以REST技術和目前最佳實務做法為基礎所建立。API為Astra部署的自動化提供基礎、並提供下列功能與優勢。

資源

共有14種REST資源類型可供使用。

API權杖存取

您可透過Astra網路使用者介面產生的API存取權杖來存取REST API。API權杖可提供對API的安全存取。

支援集合

有一組豐富的查詢參數可用來存取資源集合。部分支援的作業包括篩選、排序及分頁。

已知問題

您應該檢閱Astra Control REST API目前版本的所有已知問題。已知問題可識別可能導致您無法成功使用產品的問題。

> Astra Control REST API 22.04版本沒有新的已知問題。以下所述問題已在先前版本中發現、目前版本仍適用。

並未探索後端儲存節點中的所有儲存裝置

發出REST API呼叫以擷取儲存節點中定義的儲存裝置時、並不會傳回所有裝置。

Astra Control REST API簡介

Astra Control Center和Astra Control Service提供通用的REST API、可讓您直接透過編程語言或Curl等公用程式存取。API的主要重點和優點如下所示。

> 若要存取REST API、您必須先登入Astra網路使用者介面、然後產生API權杖。您必須在每個API要求中加入權杖。

以**REST**技術為基礎

Astra Control API是使用REST技術和目前最佳實務做法所建立。核心技術包括HTTP、Json及RBAC。

支援兩種**Astra Control**部署模式

Astra Control Service適用於公有雲環境、Astra Control Center則適用於內部部署。有一個REST API支援這兩種部署模式。

清除**REST**端點資源與物件模型之間的對應

外部REST端點、用於存取資源對應至由Astra服務內部維護的一致物件模型。物件模型是使用實體關係（ER）模型設計、有助於清楚定義API動作和回應。

豐富的查詢參數集

REST API提供一組豐富的查詢參數、可用來存取資源集合。部分支援的作業包括篩選、排序及分頁。

與**Astra Control**網路**UI**對齊

Astra網路使用者介面的設計與REST API一致、因此兩種存取路徑和使用者體驗之間保持一致。

強大的除錯與問題判斷資料

Astra Control REST API提供強大的除錯與問題判斷功能、包括系統事件與使用者通知。

工作流程程序

我們提供一組工作流程、協助您開發自動化程式碼。工作流程分為兩大類：基礎架構與管理。

先進自動化技術的基礎

除了直接存取REST API之外、您也可以使用其他以REST API為基礎的自動化技術。

Astra系列文件的一部分

Astra Control Automation文件是Astra系列文件的一部分。請參閱 "Astra文件" 以取得更多資訊。

開始使用

開始之前

您可以檢閱下列步驟、快速準備開始使用Astra Control REST API。

擁有**Astra**帳戶認證資料

您需要Astra認證資料才能登入Astra網路使用者介面並產生API權杖。使用Astra Control Center、您可以在本機管理這些認證資料。Astra Control Service可透過*驗證0*服務存取帳戶認證資料。

熟悉**Kubernetes**的基本概念

您應該熟悉幾個基本的Kubernetes概念。請參閱 "Kubernetes基本概念" 以取得更多資訊。

檢視**REST**概念與實作

請務必詳閱 "核心REST實作" 如需REST概念的相關資訊、以及Astra Control REST API的設計細節。

取得更多資訊

您應該瞭解中建議的其他資訊資源 "其他資源"。

取得**API**權杖

您需要取得Astra API權杖、才能使用Astra Control REST API。

簡介

API權杖可識別Astra的呼叫者、且必須包含在每個REST API呼叫中。

- 您可以使用Astra網路使用者介面來產生API權杖。
- 憑證隨附的使用者身分識別是由建立權杖的使用者所決定。
- 權杖必須包含在「授權」HTTP要求標頭中。
- 權杖建立後永遠不會過期。
- 您可以在Astra網路使用者介面上撤銷權杖。

相關資訊

- "撤銷API權杖"

建立**Astra API**權杖

下列步驟說明如何建立Astra API權杖。

開始之前

您需要Astra帳戶的認證資料。

關於這項工作

此工作會在Astra網路介面產生API權杖。您也應該擷取進行API呼叫時所需的帳戶ID。

步驟

1. 使用您的帳戶認證登入Astra。

   存取Astra Control Service的下列站台： "https://astra.netapp.io"

2. 按一下頁面右上角的圖示、然後選取「* API access*（* API存取*）」。
3. 按一下頁面上的「產生**API**權杖」、然後在快顯視窗中按一下「產生**API**權杖」。
4. 按一下圖示、將權杖字串複製到剪貼簿、然後儲存到編輯器中。
5. 複製並儲存相同頁面上的帳戶ID。

完成後

當您透過Curl或程式設計語言存取Astra Control REST API時、必須在HTTP「授權」要求標頭中加入API承載權杖。

您好：World

您可以在工作站的CLI上發出簡單的Curl命令、開始使用Astra Control REST API並確認其可用度。

開始之前

Curl公用程式必須可在本機工作站上使用。您也必須擁有API權杖和相關的帳戶識別碼。請參閱 "取得API權杖" 以取得更多資訊。

Curl範例

下列Curl命令會擷取Astra使用者清單。請依指示提供適當的<ACON_ID>和<API_Toke>。

```
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輸出範例

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

準備好使用工作流程

您應該先熟悉Astra工作流程的組織和格式、再將其用於即時部署。

簡介

工作流程_是完成特定管理工作或目標所需的一或多個步驟順序。Astra Control工作流程的每個步驟都是下列其中一個步驟：

- REST API呼叫（詳細資料如Curl和Json範例）
- 呼叫另一個Astra工作流程
- 其他相關工作（例如做出必要的設計決策）

工作流程包括完成每項工作所需的核心步驟和參數。這些工具可提供自訂自動化環境的起點。

通用輸入參數

以下所述的輸入參數適用於用來說明REST API呼叫的所有Curl範例。

> 由於這些輸入參數是通用需求、因此不會在個別工作流程中進一步說明。如果特定的捲曲範例使用其他輸入參數、請參閱*其他輸入參數*一節。

路徑參數

每次REST API呼叫所使用的端點路徑包括下列參數。另請參閱 "URL格式" 以取得更多資訊。

帳戶**ID**

這是UUIDv4值、可識別執行API作業的Astra帳戶。請參閱 "取得API權杖" 如需尋找帳戶ID的詳細資訊、請參閱。

要求標頭

視REST API呼叫而定、您可能需要包含數個要求標頭。

授權

工作流程中的所有API呼叫都需要API權杖來識別使用者。您必須在「授權」要求標頭中加入該標誌。請參閱 "取得API權杖" 以取得有關產生API權杖的詳細資訊。

內容類型

使用HTTP POST並將Json包含在要求本文中的要求放入、您應該根據Astra資源來宣告媒體類型。例如、您可以在建立託管應用程式的快照時、加入標題「Content-Type：application/Astra-appsnap+json」。

接受

您可以根據Astra資源、宣告回應中預期內容的特定媒體類型。例如、您可以在列出託管應用程式的備份時、加入標題「Accept: application/Astra appBackup + json」。不過為了簡化、工作流程中的捲曲範例可接受所有媒體類型。

呈現權杖和識別碼

與捲動範例搭配使用的API權杖和其他ID值不透明、沒有明顯的意義。因此為了改善範例的讀取性、不會使用實際的權杖和ID值。而是使用較小的保留關鍵字、其優點如下：

- Curl和Json樣本更清楚、更容易瞭解。
- 由於所有關鍵字的格式都與方括弧和大寫字母相同、因此您可以快速識別要插入或擷取的位置和內容。
- 不會遺失任何值、因為無法複製原始參數、並與實際部署搭配使用。

以下是Curl範例中使用的一些常用保留關鍵字。此清單並非詳盡無遺、並會視需要使用其他關鍵字。其意義應根據背景而明確。

關鍵字	類型	說明
<Account（帳戶）_ID>	路徑	UUIDv4值可識別執行API作業的帳戶。
<API_tokent>	標頭	識別及授權呼叫者的承載權杖。
<託 管應用程式ID >	路徑	UUIDv4值可識別API呼叫的託管應用程式。

工作流程類別

根據您的部署模式、Astra工作流程分為兩大類。如果您使用Astra Control Center、則應從基礎架構工作流程開始、然後繼續進行管理工作流程。使用Astra Control Service時、您通常可以直接前往管理工作流程。

工作流程中的Curl範例使用Astra Control Service的URL。當您根據環境使用內部部署的Astra Control Center時、需要變更URL。

基礎架構工作流程

這些工作流程適用於Astra基礎架構、包括認證、儲存庫和儲存後端。Astra Control Center需要這些工具、但在大多數情況下、也可搭配Astra Control Service使用。工作流程著重於建立及維護Astra託管叢集所需的工作。

管理工作流程

您可以在擁有託管叢集之後使用這些工作流程。工作流程著重於應用程式保護和支援作業、例如備份、還原及複製託管應用程式。

Kubernetes基本概念

使用Astra REST API時、有幾個Kubernetes概念是相關的。

物件

Kubernetes環境中維護的物件是代表叢集組態的持續實體。這些物件共同說明系統狀態、包括叢集工作負載。

命名空間

命名空間提供一種技術、可用來隔離單一叢集內的資源。這種組織架構在劃分工作類型、使用者和資源時非常實用。命名空間範圍為_namespac__的物件必須在命名空間內是唯一的、而具有_cluster範圍_的物件則必須在整個叢集內是唯一的。

標籤

標籤可與Kubernetes物件建立關聯。它們使用金鑰值配對來描述屬性、並可在叢集上強制執行任意組織、這對組織而言很有用、但卻不在核心Kubernetes作業範圍內。

核心REST實作

REST Web服務

代表性狀態傳輸（REST）是建立分散式Web應用程式的風格。當套用到網路服務API的設計時、它會建立一套主流技術和最佳實務做法、以揭露伺服器型資源並管理其狀態。REST為應用程式開發提供一致的基礎、但每個API的詳細資料可能會因特定設計選項而異。在使用Astra Control REST API進行即時部署之前、您應該先瞭解其特性。

資源和狀態表示

資源是網路型系統的基本元件。建立REST Web服務應用程式時、早期的設計工作包括：

- 識別系統或伺服器型資源

  每個系統都會使用及維護資源。資源可以是檔案、商業交易、程序或管理實體。根據REST Web服務設計應用程式的首要任務之一、就是識別資源。

- 資源狀態和相關狀態作業的定義

  資源永遠處於有限的狀態之一。必須清楚定義狀態、以及用來影響狀態變更的相關作業。

URI端點

每個REST資源都必須使用明確定義的定址方案來定義和提供。資源所在及識別的端點使用統一資源識別元（URI）。URI提供一般架構、可為網路中的每個資源建立唯一名稱。統一資源定位器（URL）是一種與Web服務搭配使用的URI、用於識別及存取資源。資源通常會以階層式結構公開、類似檔案目錄。

HTTP訊息

超文字傳輸傳輸協定（HTTP）是Web服務用戶端和伺服器用來交換有關資源的要求和回應訊息的傳輸協定。在設計Web服務應用程式時、HTTP方法會對應至資源及對應的狀態管理動作。HTTP為無狀態。因此、若要將一組相關的要求和回應建立關聯、以做為一筆交易的一部分、則必須在隨要求和回應資料流一起提供的HTTP標頭中加入額外資訊。

JSON格式化

雖然資訊可透過多種方式在Web服務用戶端和伺服器之間進行結構化和傳輸、但最受歡迎的選項是JavaScript物件標記法（Json）。Json是以純文字表示簡單資料結構的產業標準、用於傳輸描述資源的狀態資訊。Astra Control REST API使用Json格式化在每個HTTP要求和回應的本文中所攜帶的資料。

資源與收藏

Astra Control REST API可讓您存取資源執行個體和資源執行個體集合。

> 從概念上來說、REST 資源*類似於以物件導向程式設計（**OOP**）語言和系統所定義的*物件。有時這些術語會互換使用。但一般而言、在外部REST API的內容中使用「資源」是偏好的、而「物件」則是用於儲存在伺服器上的對應狀態執行個體資料。

Astra資源的屬性

Astra Control REST API符合RESTful設計原則。每個Astra資源執行個體都是根據明確定義的資源類型來建立。一組相同類型的資源執行個體稱為*集合*。API呼叫會對個別資源或資源集合起作用。

資源類型

Astra Control REST API隨附的資源類型具有下列特性：

- 每種資源類型都是使用架構來定義（通常是在Json中）
- 每個資源架構都包含資源類型和版本
- 資源類型是全域唯一的

資源執行個體

透過Astra Control REST API提供的資源執行個體具有下列特性：

- 資源執行個體是根據單一資源類型建立
- 資源類型會使用「媒體類型」值來表示
- 執行個體由Astra服務維護的狀態資料組成
- 每個執行個體都可透過專屬且長效的URL存取
- 如果資源執行個體可以有多種表示形式、則可以使用不同的媒體類型來要求所需的表示形式

資源集合

透過Astra Control REST API提供的資源集合具有下列特性：

- 單一資源類型的資源執行個體集稱為集合
- 資源集合具有獨特且長久存在的URL

執行個體識別碼

每個資源執行個體都會在建立時指派一個識別碼。此識別碼為128位元UUIDv4值。指派的UUUIDv4值是全域唯一且不可變的。發出API呼叫以建立新執行個體之後、會在HTTP回應的「位置」標頭中、將具有相關ID的URL傳回給呼叫者。您可以擷取識別碼、並在參照資源執行個體時用於後續通話。

> 資源識別碼是用於集合的主要金鑰。

Astra資源的通用架構

每個Astra Control資源都是使用通用結構來定義。

通用資料

每個Astra資源都包含下表所示的關鍵值。

金鑰	說明
類型	一種全域唯一的資源類型、稱為*資源類型*。
版本	稱為*資源版本*的版本識別碼。
ID	全域唯一識別碼、稱為*資源識別碼*。
中繼資料	包含各種資訊的Json物件、包括使用者和系統標籤。

中繼資料物件

每個Astra資源隨附的中繼資料Json物件包含下表所示的索引鍵值。

金鑰	說明
標籤	與資源相關聯之用戶端指定標籤的Json陣列。
建立時間戳記	Json字串包含時間戳記、指出資源建立的時間。
修改時間戳記	Json字串包含ISO-8601格式化時間戳記、指出上次變更資源的時間。
建立者	Json字串、包含建立資源之使用者ID的UUIDv4識別碼。如果資源是由內部系統元件所建立、且與建立實體沒有相關聯的UUID、則會使用* null * UUID。

資源狀態

所選資源是一種「最大」值、用於協調生命週期轉換和控制存取。

HTTP詳細資料

Astra Control REST API使用HTTP及相關參數來處理資源和集合。HTTP實作的詳細資料如下。

API交易與CRUD模式

Astra Control REST API實作交易模式、提供定義完善的作業和狀態轉換。

要求及回應API交易

每個REST API呼叫都會以HTTP要求的形式執行、以供Astra服務使用。每個要求都會產生與用戶端相關的回應。此要求回應配對可視為API交易。

支援CRUD營運模式

每個透過Astra Control REST API提供的資源執行個體和集合、都是根據* CRUD*模型來存取。共有四項作業、每項作業對應至單一HTTP方法。這些作業包括：

- 建立
- 讀取
- 更新
- 刪除

對於部分Astra資源、僅支援其中一部分作業。您應該檢閱 "API 參考" 以取得特定API呼叫的詳細資訊。

HTTP方法

下表顯示API支援的HTTP方法或動詞。

方法	CRUD	說明
取得	讀取	擷取資源執行個體或集合的物件屬性。當與集合一起使用時、這被視為* list* 作業。
貼文	建立	根據輸入參數建立新的資源執行個體。長期URL會以「位置」回應標頭傳回。
放入	更新	使用隨附的Json要求本文來更新整個資源執行個體。無法使用者修改的關鍵值會保留下來。
刪除	刪除	刪除現有的資源執行個體。

要求和回應標頭

下表摘要說明Astra Control REST API所使用的HTTP標頭。

請參閱 "RFC 7232." 和 "RFC 7233" 以取得更多資訊。

標頭	類型	使用注意事項
接受	申請	如果值為「/」或未提供、則會在內容類型回應標頭中傳回「application/json」。如果該值設為Astra資源媒體類型、則內容類型標頭會傳回相同的媒體類型。
授權	申請	含有使用者API金鑰的承載權杖。
內容類型	回應	根據「Accept（接受）」要求標頭傳回。
ETag	回應	隨附於RFC 7232.定義的成功。此值是整個Json資源之MD5值的十六進位表示法。
如果相符	申請	如3.1 RFC 72332節所述實作的先決條件要求標頭、並支援* PUT *要求。
IF修改日期	申請	如第3.4節RFC 72332所述實作的先決條件要求標頭、並支援* PUT *要求。
如果沒有修改、自此	申請	如第3.4節RFC 72332所述實作的先決條件要求標頭、並支援* PUT *要求。
位置	回應	包含新建立資源的完整URL。

查詢參數

下列查詢參數可用於資源集合。請參閱 "使用集合" 以取得更多資訊。

查詢參數	說明
包括	包含讀取集合時應傳回的欄位。
篩選器	表示讀取集合時、資源必須符合的欄位。

查詢參數	說明
訂單者	決定讀取集合時傳回資源的排序順序。
限制	限制讀取集合時傳回的資源上限。
跳過	設定讀取集合時要經過和跳過的資源數目。
數	指出中繼資料物件是否應傳回資源總數。

HTTP 狀態代碼

Astra Control REST API使用的HTTP狀態代碼如下所述。

> Astra Control REST API也使用*問題詳細資料做為HTTP API*標準。請參閱 "診斷與支援" 以取得更多資訊。

程式碼	意義	說明
200	好的	表示未建立新資源執行個體的通話成功。
201.	已建立	已成功建立物件、且位置回應標頭包含該物件的唯一識別碼。
204.	無內容	雖然未傳回任何內容、但要求仍成功。
400	錯誤要求	無法辨識或不適當的要求輸入。
401.	未獲授權	使用者未獲授權、必須經過驗證。
403.	禁止	由於授權錯誤、存取遭拒。
404..	找不到	要求中提及的資源不存在。
409.	衝突	建立物件的嘗試失敗、因為物件已經存在。
500	內部錯誤	伺服器發生一般內部錯誤。
503	服務無法使用	由於某些原因、服務尚未準備好處理要求。

URL格式

用於透過REST API存取資源執行個體或集合的URL一般結構由數個值組成。此結構反映基礎物件模型和系統設計。

帳戶為root

每個REST端點的資源路徑根目錄為Astra帳戶。因此URL中的所有路徑都以「/ACE/{ACUID_ID}」開頭、其中「ACU_ID」是帳戶的唯一UUIDv4值。內部架構反映出所有資源存取都以特定帳戶為基礎的設計。

端點資源類別

Astra資源端點分為三種不同類別：

- 核心（`/Core）
- 託管應用程式（`/k8s）
- 拓撲（`/topology）

請參閱 "資源" 以取得更多資訊。

類別版本

這三種資源類別各有一個全域版本、可控制所存取資源的版本。根據慣例和定義、移轉至資源類別的新主要版本（例如、從「/v1」移至「/v2」）、將會導致API發生突破性變更。

資源執行個體或集合

根據資源執行個體或集合是否被存取、路徑中可以使用資源類型和識別碼的組合。

範例

- 資源路徑

  根據上述結構、端點的典型路徑為：「/Accounts / ｛Account _id｝/核心/ v1 /使用者」。

- 完整URL

  對應端點的完整URL為：「https://astra.netapp.io/accounts/{account_id}/core/v1/users`。

資源與端點

您可以使用Astra Control REST API提供的資源、將Astra部署自動化。每個資源都是透過一或多個端點進行存取。以下資訊將介紹您可在自動化部署中使用的REST資源。

> 用於存取Astra Control資源的路徑格式和完整URL是以數個值為基礎。請參閱 "URL格式" 以取得更多資訊。另請參閱 "API 參考" 如需使用Astra資源和端點的詳細資訊、

Astra Control REST資源摘要

Astra Control REST API中提供的主要資源端點分為三類。每個資源都可以使用CRUD作業的完整集合（建立、讀取、更新、刪除）存取、除非另有說明。

「版本」欄位會指出首次引進資源時的Astra版本。此欄位會以粗體顯示新增至目前版本的資源。

核心資源

核心資源端點提供建立及維護Astra執行時間環境所需的基礎服務。

資源	版本	說明
帳戶	21：12	帳戶資源可讓您管理多租戶Astra Control部署環境中的隔離租戶。
ASUP	21.08年	ASUP資源代表AutoSupport 轉寄給NetApp支援部門的各種產品組合。
認證資料	21.04.	認證資源包含安全性相關資訊、可與Astra使用者、叢集、儲存區及儲存後端搭配使用。
權利	21.08年	權利資源是根據使用中的授權與訂閱、代表帳戶可用的功能與容量。
活動	21.04.	事件資源代表系統中發生的所有事件、包括分類為通知的子集。
執行掛勾	21：12	執行攔截資源代表自訂指令碼、您可以在執行託管應用程式快照之前或之後執行。
功能	21.08年	功能資源代表所選的Astra功能、您可以查詢這些功能、以判斷系統中是否已啟用或停用這些功能。存取限制為唯讀。
攔截來源	21：12	Hook來源資源代表實際使用的原始程式碼與執行掛勾。將原始程式碼與執行控制區分開有幾項優點、例如允許共用指令碼。
授權	21.08年	授權資源代表Astra帳戶可用的授權。
通知	21.04.	通知資源代表有通知目的地的Astra事件。存取權限是以每位使用者為單位提供。
套件	* 22.04 *	套件資源提供套件定義的登錄與存取。軟體套件包含各種元件、包括檔案、映像及其他成品。
角色繫結	21.04.	角色繫結資源代表特定使用者與帳戶配對之間的關係。除了兩者之間的連結之外、還會透過特定角色為每個角色指定一組權限。
設定	21.08年	設定資源代表一組金鑰值配對、說明特定Astra帳戶的功能。
訂購	21.08年	訂閱資源代表Astra帳戶的使用中訂閱。

資源	版本	說明
權杖	21.04.	權杖資源代表可以程式設計方式存取Astra Control REST API的權杖。
未讀取通知	21.04.	未讀取的通知資源代表指派給特定使用者但尚未讀取的通知。
升級	* 22.04 *	升級資源可存取軟體元件、以及啟動升級的能力。
使用者	21.04.	使用者資源代表Astra使用者能夠根據其定義的角色來存取系統。

託管應用程式資源

託管應用程式資源端點可讓您存取託管的Kubernetes應用程式。

資源	版本	說明
應用程式資產	21.04.	應用程式資產資源代表管理Astra應用程式所需的狀態資訊內部集合。
應用程式備份	21.04.	應用程式備份資源代表託管應用程式的備份。
應用程式快照	21.04.	應用程式快照資源代表託管應用程式的快照。
執行攔截置換	21：12	執行掛勾置換資源可讓您視需要停用特定應用程式的預先載入NetApp預設執行掛勾。
託管應用程式	21.04.	託管應用程式資源代表由Astra管理的Kubernetes應用程式。
排程	21.04.	排程資源代表排程用於託管應用程式的資料保護作業、做為資料保護原則的一部分。

拓撲資源

拓撲資源端點可讓您存取未受管理的應用程式和儲存資源。

資源	版本	說明
應用程式	21.04.	應用程式資源代表Kubernetes的所有應用程式、包括Astra未管理的應用程式。
鏟斗	21.08年	儲存庫資源代表S3雲端儲存庫、用於儲存由Astra管理之應用程式的備份。
雲端	21.08年	雲端資源代表Astra用戶端可連線至的雲端、以便管理叢集和應用程式。
叢集	21.08年	叢集資源代表Kubernetes叢集、並非由Kubernetes管理。
叢集節點	21：12	叢集節點資源可讓您存取Kubernetes叢集中的個別節點、以提供額外的解析度。
託管叢集	21.08年	託管叢集資源代表Kubernetes目前由Kubernetes管理的叢集。
託管儲存後端	21：12	託管儲存後端資源可讓您存取後端儲存供應商的抽象呈現。這些儲存後端可由託管叢集和應用程式使用。
命名空間	21：12	命名空間資源可讓您存取Kubernetes叢集中所使用的命名空間。
儲存後端	21.08年	儲存後端資源代表可由Astra託管叢集和應用程式使用的儲存服務供應商。
儲存類別	21.08年	儲存類別資源代表所探索到的不同類別或類型的儲存設備、可供特定的託管叢集使用。
Volume	21.04.	Volume資源代表與託管應用程式相關聯的Kubernetes儲存磁碟區。

最新版本的新端點

目前的22.04 Astra Control版本已新增下列REST端點。此外、多個現有資源的版本也已升級。

- /Accounts / {account_id} /核心/ v1/packages
- /Accounts / {account_id} /core / v1/packages/ {package_id}
- /Accounts / {account_id} /核心/ v1/升級
- /Accounts / {account_id} /核心/ v1/exgrades// {grade_id}
- /Accounts / {account_id} /拓撲/ v1/appBackups
- /Accounts / {account_id} /拓撲/ v1/appBackups/ {appBackup}
- /Accounts / {account_id} /拓撲/ v1/v雲端/ {雲端_id} /叢集/ {cluster_id} /叢集節點
- /Accounts / {account_id} /拓撲/ v1/v雲端/ {雲端_id} /叢集/ {cluster_id} /叢集節點/ {clusterNode_id}
- /accounts/{account_id}/topology / v1/managedClusters/ {managedCluster_id} /apps / {app_id } /appAsset
- /Accounts / {account_id} /拓撲/ v1/managedClusters/ {managedCluster_id} /應用程式/ {app_id} /應用程式資產/ {appAsset_id}
- /accounts/{account_id}/topology / v1/managedClusters/{managedCluster_id}/clusterNode
- /Accounts / {account_id} /拓撲/ v1/managedClusters/ {managedCluster_id} /叢集節點/ {clusterNode_id }

其他資源和端點

您可以使用數種額外的資源和端點來支援Astra部署。

Astra Control REST API參考文件目前未隨附這些資源和端點。

OpenAPI

OpenAPI端點可讓您存取目前的OpenAPI Json文件及其他相關資源。

OpenMetrics

OpenMetrics端點可透過OpenMetrics資源存取帳戶指標。Astra Control Center部署模式可提供支援。

其他使用考量

RBAC安全性

Astra REST API支援角色型存取控制（RBAC）、以限制系統功能的存取。

Astra角色

每位Astra使用者都會被指派到單一角色、以決定可執行的動作。這些角色會依照下表所述的階層架構來排列。

角色	說明
擁有者	擁有管理員角色的所有權限、也可以刪除Astra帳戶。
管理	擁有成員角色的所有權限、也可以邀請使用者加入帳戶。
成員	能夠完全管理Astra應用程式和運算資源。
檢視者	僅限檢視資源。

增強的RBAC與命名空間精細度

此功能是隨Astra REST API 22.04版一起推出。

為特定使用者建立角色繫結時、可以套用限制來限制使用者可存取的命名空間。有幾種方法可以定義此限制、如下表所述。如需詳細資訊、請參閱角色繫結API中的「角色繫結」參數。

命名空間	說明
全部	使用者可透過萬用字元參數「*」存取所有命名空間。這是維持回溯相容性的預設值。
無	限制清單是空的、但仍會指定。這表示使用者無法存取任何命名空間。
命名空間清單	命名空間的UUID會包含在內、限制使用者只能使用單一命名空間。您也可以使用以逗號分隔的清單來存取多個命名空間。
標籤	系統會指定標籤、並允許存取所有相符的命名空間。

使用集合

Astra Control REST API提供數種不同的方法、可透過定義的查詢參數來存取資源集合。

選擇值

您可以使用「include」參數、指定每個資源執行個體應傳回的金鑰值配對。所有執行個體都會傳回回應本文中。

篩選

集合資源篩選可讓API使用者指定條件、以判斷回應本文中是否傳回資源。使用「filter」參數來表示篩選條件。

排序

集合資源排序可讓API使用者指定資源在回應本文中的傳回順序。使用「OrderBy」參數來表示篩選條件。

分頁

您可以使用「限制」參數、限制在要求時傳回的資源執行個體數目、以強制分頁。

數

如果您將布林參數「counts」設為「true」、則會在中繼資料區段中提供傳回之給定回應陣列中的資源數量。

診斷與支援

Astra Control REST API提供多項支援功能、可用於診斷與偵錯。

API資源

API資源提供多項Astra功能、可提供診斷資訊和支援。

類型	說明
活動	在Astra處理過程中記錄的系統活動。
通知	被視為重要的事件子集、足以呈現給使用者。
未讀取通知	使用者尚未讀取或擷取的通知。

撤銷API權杖

您可以在Astra Web介面上撤銷不再需要的API權杖。

開始之前

您需要Astra帳戶。您也應該識別要撤銷的權杖。

關於這項工作

撤銷權杖之後、權杖會立即且永久地無法使用。

步驟

1. 使用您的帳戶認證登入Astra。

   存取Astra Control Service的下列站台： "https://astra.netapp.io"

2. 按一下頁面右上角的圖示、然後選取「* API access*（* API存取*）」。
3. 選取您要撤銷的權杖。
4. 在「動作」下拉式方塊下、按一下「撤銷權杖」。

基礎架構工作流程

開始之前

您可以使用這些工作流程來建立及維護Astra Control Center部署模式所使用的基礎架構。在大多數情況下、工作流程也可搭配Astra Control Service使用。

> NetApp可隨時擴充及強化這些工作流程、因此您應定期檢閱。

一般準備

使用任何Astra工作流程之前、請務必先檢閱 "準備好使用工作流程"。

工作流程類別

基礎架構工作流程會依不同類別進行組織、以便更容易找到您想要的工作流程。

類別	說明
身分識別與存取	這些工作流程可讓您管理身分識別、以及Astra的存取方式。這些資源包括使用者、認證和權杖。
桶	您可以使用這些工作流程來建立及管理用來儲存備份的S3儲存區。
儲存設備	這些工作流程可讓您新增及維護儲存後端和磁碟區。
叢集	您可以新增受管理的Kubernetes叢集、以保護及支援其中所包含的應用程式。

身分識別與存取

列出使用者

您可以列出針對特定Astra帳戶所定義的使用者。

1.列出使用者

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/｛AccountID｝/核心/ v1/uss

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
包括	查詢	否	選擇性地選取您要傳回回應中的值。

Curl範例：傳回所有使用者的所有資料

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

Curl範例：傳回所有使用者的名字、姓氏和ID

```
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輸出範例

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

桶

列出庫存箱

您可以列出針對特定Astra帳戶所定義的S3儲存區。

1.列出庫存箱

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/topology / v1/buckets

Curl範例：傳回所有儲存區的所有資料

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

儲存設備

列出儲存後端

您可以列出可用的儲存後端。

1.列出庫存箱

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/topology / v1/storageBackends

Curl範例：傳回所有儲存後端的所有資料

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

Json輸出範例

```
{
  "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"
    }
  ]
}
```


叢集

列出託管叢集

您可以列出目前由Astra管理的Kubernetes叢集。

1.列出叢集

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/topology / v1/managedClusters

Curl範例：傳回所有叢集的所有資料

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

管理工作流程

開始之前

您可以使用這些工作流程來管理Astra託管叢集內的應用程式。

> NetApp可隨時擴充及強化這些工作流程、因此您應定期檢閱。

一般準備

使用任何Astra工作流程之前、請務必先檢閱 "準備好使用工作流程"。

工作流程類別

管理工作流程會依不同類別進行組織、以便更容易找到您想要的工作流程。

類別	說明
應用程式控制	這些工作流程可讓您控制託管及未受管理的應用程式。您可以列出應用程式、以及建立和移除託管應用程式。
應用程式保護	您可以使用這些工作流程、透過快照和備份來保護託管應用程式。
複製及還原應用程式	這些工作流程說明如何複製及還原託管應用程式。
支援	有幾種工作流程可用來偵錯及支援您的應用程式、以及一般Kubernetes環境。

其他考量

使用管理工作流程時、還有幾項額外考量。

複製應用程式

複製應用程式時、需要考量一些事項。下列參數是Json輸入的一部分。

來源叢集識別碼
: 「資源叢集ID」的值一律會識別安裝原始應用程式的叢集。

叢集識別碼
: 「clusterid」的值表示要安裝新應用程式的叢集。

- 在同一個叢集內進行複製時、「clusterid」和「資源叢集ID」的值相同。
- 在叢集之間進行複製時、兩個值不同、而「clusterid」應為目標叢集的ID。

命名空間
: 「命名空間」值必須與原始來源應用程式不同。此外、複本的命名空間也不存在、Astra也會建立該名稱空間。

備份與快照

您可以選擇使用「backupID」或「snapshotID」參數、從現有的備份或快照複製應用程式。如果您未提供備份或快照、Astra會先建立應用程式的備份、然後從備份複製。

還原應用程式

以下是還原應用程式時的幾項考量事項。

- 還原應用程式與複製作業非常類似。
- 還原應用程式時、您必須提供備份或快照。

應用程式控制

列出未受管理的應用程式

您可以列出目前未由Astra管理的應用程式。您可以在選擇要管理的應用程式時執行此動作。

依預設、這些工作流程中使用的REST端點會傳回所有Astra應用程式。您可以在API呼叫上使用「filter」查詢參數、只要求傳回未受管理的應用程式。您也可以省略filter參數來傳回所有應用程式、然後檢查輸出中的「managedState」欄位、以判斷哪些應用程式處於「Unmanaged」狀態。

僅列出管理狀態等於**Unmanaged**的應用程式

此工作流程使用「filter」查詢參數、只傳回未受管理的應用程式。

1.列出未受管理的應用程式

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/topology / v1/apps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
篩選器	查詢	否	使用篩選器來指定應傳回哪些應用程式。
包括	查詢	否	選擇性地選取您要傳回回應中的值。

Curl範例：傳回未受管理應用程式的名稱、**ID**和管理狀態

```
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輸出範例


```
{
  "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": {}
}
```

列出所有應用程式、然後選取未受管理的應用程式

此工作流程會傳回所有應用程式。您必須檢查輸出、判斷哪些是未受管理的。

1.列出所有應用程式

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/topology / v1/apps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
包括	查詢	否	選擇性地選取您要傳回回應中的值。

Curl範例：傳回所有應用程式的所有資料

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

Curl範例：傳回所有應用程式的名稱、**ID**和管理狀態

```
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輸出範例

```
{
  "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": {}
}
```

2.選取未受管理的應用程式

檢閱API呼叫的輸出、然後手動選取「managedState」等於「Unmanaged」的應用程式。

列出託管應用程式

您可以列出目前由Astra管理的應用程式。您可以在尋找特定應用程式的快照或備份時執行此動作。

1.列出應用程式

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/{AccountID}/k8s/v1/managedApps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
包括	查詢	否	選擇性地選取您要傳回回應中的值。

Curl範例：傳回所有應用程式的所有資料

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

Curl範例：傳回所有應用程式的名稱、**ID**和狀態

```
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輸出範例

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

取得託管應用程式

您可以擷取所有描述單一託管應用程式的資源變數。

開始之前

您必須擁有想要擷取的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

1.取得應用程式

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id}

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	要擷取之託管應用程式的ID值。

Curl範例：傳回應用程式的所有資料

```
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>'
```

管理應用程式

您可以根據Astra已知的應用程式來建立託管應用程式。管理應用程式時、您可以定期備份

和快照來保護應用程式。

開始之前

您必須擁有要管理的探索應用程式ID。如有需要、您可以使用工作流程 "列出未受管理的應用程式" 以找出應用程式。

1.管理應用程式

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Account/{AccountID}/k8s/v1/managedApps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
JSON	本文	是的	提供識別要管理之應用程式所需的參數。請參閱以下範例。

JSONN輸入範例

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

Curl範例：管理應用程式

```
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
```

取消管理應用程式

您可以在不再需要的情況下移除託管應用程式。移除託管應用程式也會刪除相關的排程。

開始之前

您必須擁有想要取消管理的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

應用程式的備份與快照不會在刪除時自動移除。如果您不再需要備份和快照、則應在移除應用程式之前先將其刪除。

1.不受管理的應用程式

執行下列REST API呼叫。

HTTP方法	路徑
刪除	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id}

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別要移除的託管應用程式。

Curl範例：移除託管應用程式

```
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>'
```

應用程式保護

列出快照

您可以列出針對特定託管應用程式所拍攝的快照。

開始之前

您必須擁有想要列出快照的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

1.列出快照

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /應用程式快照

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別擁有所列快照的託管應用程式。
數	查詢	否	如果是「count=true」、快照數量會包含在回應的中繼資料區段中。

Curl範例：傳回應用程式的所有快照

```
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範例：傳回應用程式和計數的所有快照

```
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輸出範例

```
{
  "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
  }
}
```


列出備份

您可以列出已為特定託管應用程式建立的備份。

開始之前

您必須擁有要列出備份的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

1.列出備份

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /appBackups

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別擁有所列備份的託管應用程式。

Curl範例：傳回應用程式的所有備份

```
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輸出範例

```
{
  "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": {}
}
```

為託管應用程式建立快照

您可以為特定的託管應用程式建立快照。

開始之前

您必須擁有想要建立快照的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

1.建立快照

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /應用程式快照

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別要建立快照的託管應用程式。
JSON	本文	是的	提供快照的參數。請參閱以下範例。

JSONN輸入範例

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

Curl範例：建立應用程式的快照

```
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
```

為託管應用程式建立備份

您可以為特定的託管應用程式建立備份。您可以使用備份來還原或複製應用程式。

開始之前

您必須擁有想要建立備份的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。

1.建立備份

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /appBackups

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別要建立備份的託管應用程式。
JSON	本文	是的	提供備份參數。請參閱以下範例。

JSONN輸入範例

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

Curl範例：為應用程式建立備份

```
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
```

刪除快照

您可以刪除與託管應用程式相關的快照。

開始之前

您必須具備下列條件：

- 擁有快照的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。
- 您要刪除的快照ID。如有需要、您可以使用工作流程 "列出快照" 以找出快照。

1.刪除快照

執行下列REST API呼叫。

HTTP方法	路徑
刪除	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /應用程式快照/ {appsnap_id}

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別擁有快照的託管應用程式。
Snapshot ID	路徑	是的	識別要刪除的快照。

Curl範例：刪除應用程式的單一快照

```
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>'
```

刪除備份

您可以刪除與託管應用程式相關的備份。

開始之前

您必須具備下列條件：

- 擁有備份的託管應用程式ID。如有需要、您可以使用工作流程 "列出託管應用程式" 以找出應用程式。
- 您要刪除的備份ID。如有需要、您可以使用工作流程 "列出備份" 以找出快照。

1.刪除備份

執行下列REST API呼叫。

> 您可以使用選用的要求標頭強制刪除失敗的備份、如下所述。

HTTP方法	路徑
刪除	/Accounts / {account_id} /k8s/v1/managedApps/ {managedApp_id} /appBackups/ {appBackup}

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
託管應用程式ID	路徑	是的	識別擁有備份的託管應用程式。
備份ID	路徑	是的	識別要刪除的備份。
強制刪除	標頭	否	用於強制刪除失敗的備份。

Curl範例：刪除應用程式的單一備份

```
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範例：使用**force**選項刪除應用程式的單一備份

```
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'
```

複製及還原應用程式

複製託管應用程式

您可以透過複製現有的託管應用程式來建立新的應用程式。

開始之前

請注意下列關於此工作流程的資訊：

- 未使用應用程式備份或快照
- 複製作業會在同一個叢集內執行

> 若要將應用程式複製到不同的叢集、您必須視環境的需求、更新Json輸入中的「clusterid」參數。

1.選取要複製的託管應用程式

執行工作流程 "列出託管應用程式" 並選取您要複製的應用程式。用於複製應用程式的REST呼叫需要幾個資源值。

2.複製應用程式

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Account/{AccountID}/k8s/v1/managedApps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
JSON	本文	是的	提供複製應用程式的參數。請參閱以下範例。

JSONN輸入範例

```
{
  "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"
}
```

Curl範例：複製應用程式

```
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
```

從快照複製託管應用程式

您可以從應用程式快照複製新應用程式、以建立新的應用程式。

開始之前

請注意下列關於此工作流程的資訊：

- 使用應用程式快照
- 複製作業會在同一個叢集內執行

> 若要將應用程式複製到不同的叢集、您必須視環境的需求、更新Json輸入中的「clusterid」參數。

1.選取要複製的託管應用程式

執行工作流程 "列出託管應用程式" 並選取您要複製的應用程式。用於複製應用程式的REST呼叫需要幾個資源值。

2.選取要使用的快照

執行工作流程 "列出快照" 然後選取您要使用的快照。

3.複製應用程式

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Account/{AccountID}/k8s/v1/managedApps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
JSON	本文	是的	提供複製應用程式的參數。請參閱以下範例。

JSONN輸入範例

```
{
  "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範例：從快照複製應用程式

```
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
```

從備份複製託管應用程式

您可以從應用程式備份複製新的託管應用程式、藉此建立新的託管應用程式。

開始之前

請注意下列關於此工作流程的資訊：

- 使用應用程式備份
- 複製作業會在同一個叢集內執行

> 若要將應用程式複製到不同的叢集、您必須視環境的需求、更新Json輸入中的「clusterid」參數。

1.選取要複製的託管應用程式

執行工作流程 "列出託管應用程式" 並選取您要複製的應用程式。用於複製應用程式的REST呼叫需要幾個資源值。

2.選取要使用的備份

執行工作流程 "列出備份" 並選取您要使用的備份。

3.複製應用程式

執行下列REST API呼叫。

HTTP方法	路徑
貼文	/Account/{AccountID}/k8s/v1/managedApps

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
JSON	本文	是的	提供複製應用程式的參數。請參閱以下範例。

JSONN輸入範例

```
{
  "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範例：從備份複製應用程式

```
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
```

從備份還原託管應用程式

您可以從備份建立新的應用程式來還原託管應用程式。

1.選取要還原的託管應用程式

執行工作流程 "列出託管應用程式" 並選取您要複製的應用程式。用於複製應用程式的REST呼叫需要幾個資源值。

2.選取要使用的備份

執行工作流程 "列出備份" 並選取您要使用的備份。

3.還原應用程式

執行下列REST API呼叫。您必須提供備份（如下所示）或快照的ID。

HTTP方法	路徑
放入	/Account/{AccountID}/k8s/v1/managedApps/{AppID}

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
JSON	本文	是的	提供複製應用程式的參數。請參閱以下範例。

JSONN輸入範例

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

Curl範例：從備份中就地還原應用程式

```
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
```

支援

列出通知

您可以列出特定Astra帳戶的通知。您可以在監控系統活動或偵錯問題時執行此動作。

1.列出通知

執行下列REST API呼叫。

HTTP方法	路徑
取得	/Account/｛AccountID｝/核心/ v1/notifications

其他輸入參數

除了所有REST API呼叫通用的參數之外、此步驟的Curl範例也會使用下列參數。

參數	類型	必要	說明
篩選器	查詢	否	選擇性地篩選您要在回應中傳回的通知。
包括	查詢	否	選擇性地選取您要傳回回應中的值。

Curl範例：傳回所有通知

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

Curl範例：傳回嚴重性為警告的通知說明

```
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輸出範例

```
{
  "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": {}
}
```

刪除失敗的應用程式

如果託管應用程式的備份或快照處於故障狀態、您可能無法移除該應用程式。在此情況下、您可以使用下列工作流程手動移除應用程式。

1.選取要刪除的託管應用程式

執行工作流程 "列出託管應用程式" 並選取您要移除的應用程式。

2.列出應用程式的現有備份

執行工作流程 "列出備份"。

3.刪除所有備份

執行工作流程、刪除所有應用程式備份 "刪除備份" 針對清單中的每個備份。

4.列出應用程式的現有快照

執行工作流程 "列出快照"。

5.刪除所有快照

執行工作流程 "刪除快照" 從清單中的每個快照。

6.移除應用程式

執行工作流程 "取消管理應用程式" 移除應用程式。

使用Python

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK是開放原始碼套件、可用於自動化Astra Control部署。此套件也是瞭解Astra Control REST API的寶貴資源、可能是建立自己自動化平台的一部分。

> 為了簡化作業、NetApp Astra Control Python SDK將在本頁的其餘部分中稱為* SDK*。

兩種相關軟體工具

SDK包含兩種不同但相關的工具、可在存取Astra Control REST API時以不同的抽象層級運作。

Astra SDK

Astra SDK提供核心平台功能。其中包含一組Python類別、可抽象化基礎REST API呼叫。這些類別可支援各種Astra Control資源的管理動作、包括應用程式、備份、快照和叢集。

Astra SDK是套件的一部分、以單一的「astrasdk.py」檔案提供。您可以將此檔案匯入您的環境、並直接使用類別。

> * NetApp Astra Control Python SDK*（或僅SDK）是整個套件的名稱。* Astra SDK*是指單一檔案「astrasdk.py」中的核心Python類別。

工具套件指令碼

除了Astra SDK檔案之外、也提供「toolkit.py」指令碼。此指令碼可存取內部定義為Python功能的獨立管理動作、以更高的抽象層級運作。指令碼會匯入Astra SDK、並視需要呼叫類別。

如何存取

您可以使用下列方式存取SDK。

Python套件

SDK可從取得 "Python套件索引" 以* NetApp-Astra工具套件*的名稱。套件已指派版本編號、並將視需要繼續更新。您必須使用*子母畫面*套件管理公用程式、將套件安裝到您的環境中。

請參閱 "PyPI：NetApp Astra Control Python SDK" 以取得更多資訊。

GitHub原始程式碼

您也可以在GitHub取得SDK原始程式碼。儲存庫包含下列項目：

- 「astrasdk.py」（使用Python類別的Astra SDK）
- 「toolit.py」（較高層級的功能型指令碼）
- 詳細的安裝要求與指示
- 安裝指令碼
- 其他文件

您可以複製 "GitHub：NetApp/NetApp-Astra工具套件" 儲存庫。

安裝與基本需求

在安裝套件及準備使用套件時、需要考量幾種選項和需求。

安裝選項摘要

您可以使用下列其中一種方法來安裝SDK：

- 使用Pip將套件從PyPI安裝到Python環境
- 複製GIT Hub儲存庫、然後：
  - 將套件部署為Docker容器（包含您所需的一切）
  - 複製兩個核心Python檔案、以便Python用戶端程式碼存取

如需詳細資訊、請參閱PyPI和GitHub頁面。

Astra Control環境的需求

無論是直接使用Astra SDK中的Python類別、或是「toolkit.py」指令碼中的功能、最終都能在Astra Control部署中存取REST API。因此、您需要Astra帳戶和API權杖。請參閱 "開始之前" 如需詳細資訊、請參閱本文件*入門*一節中的其他頁面。

NetApp Astra Control Python SDK的需求

SDK有幾項與本機Python環境相關的先決條件。例如、您必須使用Python 3.5或更新版本。此外、還需要幾個Python套件。如需詳細資訊、請參閱GitHub儲存庫頁面或PyPI套件頁面。

實用資源摘要

以下是您開始使用所需的一些資源。

- "PyPI：NetApp Astra Control Python SDK"
- "GitHub：NetApp/NetApp-Astra工具套件"

原生Python

開始之前

Python是熱門的開發語言、尤其是資料中心自動化。在將Python的原生功能與數個通用套件搭配使用之前、您必須先準備環境和所需的輸入檔。

> 除了直接使用Python存取Astra Control REST API之外、NetApp也提供工具套件、可擷取API並移除部分複雜性。請參閱 "NetApp Astra Control Python SDK" 以取得更多資訊。

準備環境

執行Python指令碼的基本組態需求如下所述。

Python 3.

您需要安裝最新版本的Python 3。

其他程式庫

必須安裝*請求*和* urllib3*程式庫。您可以根據環境使用pip或其他Python管理工具。

網路存取

執行指令碼的工作站必須具備網路存取權、而且必須能夠連線到Astra Control。使用Astra Control Service時、您必須連線至網際網路、才能連線至https://astra.netapp.io`上的服務。

身分識別資訊

您需要具有帳戶識別碼和API權杖的有效Astra帳戶。請參閱 "取得API權杖" 以取得更多資訊。

建立**Json**輸入檔

Python指令碼仰賴Json輸入檔中包含的組態資訊。範例檔案如下所示。

> 您需要視環境的需求更新範例。

身分識別資訊

下列檔案包含API權杖和Astra帳戶。您需要使用「-I」（或「-identity」）CLI參數、將此檔案傳遞給Python指令碼。

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

列出託管應用程式

您可以使用下列指令碼來列出Astra帳戶的託管應用程式。

> 請參閱 "開始之前" 以取得所需Json輸入檔的範例。

```
#!/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 參考

您可以存取所有Astra Control REST API呼叫的詳細資料、包括HTTP方法、輸入參數和回應。這份完整參考資料有助於使用REST API開發自動化應用程式。

REST API參考文件目前已隨Astra Control提供、並可線上取得。

開始之前

您需要Astra Control Center或Astra Control Service帳戶。

步驟

1. 使用您的帳戶認證登入Astra。

   存取Astra Control Service的下列站台： "https://astra.netapp.io"

2. 按一下頁面右上角的圖示、然後選取「* API access*（* API存取*）」。
3. 在頁面頂端、按一下「* API Documentation （API文件*）」下方顯示的URL。
4. 出現提示時、請再次提供您的帳戶認證資料。

其他資源

您可以存取更多資源、以取得協助、並尋找更多有關NetApp雲端服務與支援、以及一般REST與雲端概念的資訊。

Astra

- "Astra Control Center 22.04文件"

  部署在客戶駐地的Astra Control Center軟體目前版本的文件。

- "Astra Control Service文件"

  目前在公有雲上提供Astra Control Service軟體版本的文件。

- "Astra Trident文件"

  由NetApp維護的開放原始碼儲存協調程式Astra Trident軟體目前版本的文件。

- "Astra系列文件"

  集中位置、可存取內部部署和公有雲部署的所有Astra文件。

NetApp雲端資源

- "NetApp雲端解決方案"

  NetApp雲端解決方案的中央網站。

- "NetApp Cloud Central主控台"

  使用登入功能的NetApp Cloud Central服務主控台。

- "NetApp支援"

  存取疑難排解工具、文件及技術支援協助。

REST與雲端概念

- 博士 "論文" 作者：Roy Fielding

  本出版品介紹並建立REST應用程式開發模式。

- "驗證0"

  這是Astra服務用於網路存取的驗證與授權平台服務。

- "RFC編輯器"

Web和網際網路標準的權威來源、以獨特編號的RFC文件集合形式加以維護。

舊版Astra Control Automation文件

您可以從下列連結存取先前Astra Control版本的自動化文件。

- "Astra Control Automation 21.12文件"
- "Astra Control Automation 21.08文件"

法律聲明

法律聲明提供版權聲明、商標、專利等存取權限。

版權

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

商標

NetApp 、 NetApp 標誌及 NetApp 商標頁面上列出的標章均為 NetApp 、 Inc. 的商標。其他公司與產品名稱可能為其各自所有者的商標。

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

專利

如需最新的 NetApp 擁有專利清單、請參閱：

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

隱私權政策

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

Astra Control API授權

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