Astra Control Automation 22.04 - Dokumentation

Astra Automation 22.04

NetApp
June 28, 2024

Inhalt

Astra Control Automation 22.04 - Dokumentation

Versionshinweise

Informationen zu diesem Release

Die Dokumentation auf dieser Website beschreibt die Astra Control REST API und damit verbundene Automatisierungstechnologien, die mit der April 2022 (22.04) Version von Astra Control verfügbar sind. Diese Version der REST-API ist insbesondere mit den entsprechenden 22.04 Versionen von Astra Control Center und Astra Control Service enthalten.

Weitere Informationen zu diesem Release sowie zu vorherigen Versionen finden Sie auf den folgenden Seiten und den folgenden Websites:

- "Neuerungen bei der Astra Control REST-API"
- "REST-Ressourcen und -Endpunkte"
- "Astra Control Center 22.04-Dokumentation"
- "Dokumentation des Astra Control Service"
- "Frühere Versionen der Dokumentation von Astra Automation"

Was ist neu mit der Astra Control REST API

NetApp aktualisiert regelmäßig die Astra Control REST API und bietet Ihnen neue Funktionen, Verbesserungen und Fehlerbehebungen.

26. April 2022 (22.04)

Diese Version umfasst eine Erweiterung und Aktualisierung der REST-API sowie erweiterte Sicherheits- und Administrationsfunktionen.

Neue und verbesserte Astra-Ressourcen

Es wurden zwei neue Ressourcen-Typen hinzugefügt: **Paket** und **Upgrade**. Außerdem wurden die Versionen verschiedener vorhandener Ressourcen aktualisiert.

Erweiterte RBAC mit Namespace-Granularität

Wenn Sie eine Rolle einem zugeordneten Benutzer zuweisen, können Sie die Namespaces beschränken, auf die der Benutzer Zugriff hat. Siehe * Role Binding API* Referenz und "RBAC-Sicherheit" Finden Sie weitere Informationen.

Entfernen des Buckets

Sie können einen Eimer entfernen, wenn er nicht mehr benötigt wird oder nicht ordnungsgemäß funktioniert.

Unterstützung von Cloud Volumes ONTAP

Cloud Volumes ONTAP wird nun als Storage Back-End unterstützt.

Zusätzliche Produktverbesserungen

Die beiden Astra Control-Produktimplementierungen sind mit einigen zusätzlichen Verbesserungen vertraut:

- Generischer Eingang für Astra Control Center
- Privates Cluster in AKS
- Unterstützung für Kubernetes 1.22
- Unterstützung des VMware Tanzu Portfolios

Sehen Sie sich die Seite **Was ist neu** auf den Dokumentationsseite des Astra Control Centers und des Astra Control Service an.

Verwandte Informationen

- "Astra Control Center: Was ist neu"
- "Astra Control Service: Was ist neu"

Bis 14. Dezember 2021 (21.12)

Dieses Release enthält eine Erweiterung der REST API sowie eine Änderung der Dokumentationsstruktur, um die Entwicklung von Astra Control durch zukünftige Release-Updates besser zu unterstützen.

Separate Dokumentation für Astra Automation für jede Version von Astra Control

Jede Version von Astra Control verfügt über eine eigene REST-API, die auf die Funktionen der spezifischen Version zugeschnitten wurde. Die Dokumentation für jede Version der Astra Control REST API ist jetzt auf einer eigenen dedizierten Website zusammen mit dem zugehörigen GitHub Content Repository verfügbar. Die Hauptdoktorandseite "Astra Control Automation" Enthält immer die Dokumentation für die aktuellste Version. Siehe "Frühere Versionen der Dokumentation Astra Control Automation" Weitere Informationen zu vorherigen Releases.

Erweiterung der REST-Ressourcentypen

Die Anzahl DER REST-Ressourcentypen hat sich mit Schwerpunkt auf Ausführungs-Hooks und Storage-Back-Ends weiter erweitert. Die neuen Ressourcen umfassen: Konto, Testsuite, Hook Source, Execution Hook Override, Cluster Node, Managed Storage Back-End, Namespace, Storage-Gerät und Storage-Node. Siehe "Ressourcen" Finden Sie weitere Informationen.

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK ist ein Open-Source-Paket, mit dem sich der Automatisierungscode für Ihre Astra Control Umgebung leichter entwickeln lässt. Der Kern ist das Astra SDK, das eine Reihe von Klassen umfasst, um die Komplexität der REST API Aufrufe zu abstrahieren. Es gibt auch ein Toolkit-Skript zur Ausführung spezifischer administrativer Aufgaben durch Zusammenfassung und Abstrahierung der Python-Klassen. Siehe "NetApp Astra Control Python SDK" Finden Sie weitere Informationen.

August 5 2021 (21.08)

Diese Version umfasst die Einführung eines neuen Astra Implementierungsmodells und eine wesentliche Erweiterung der REST-API.

Astra Control Center-Implementierungsmodell

Neben dem vorhandenen Astra Control Service, der als Public Cloud-Service bereitgestellt wird, umfasst diese Version auch das On-Premises-Implementierungsmodell von Astra Control Center. Sie können Astra Control Center an Ihrem Standort installieren und so Ihre lokale Kubernetes-Umgebung managen. Die beiden Astra Control Implementierungsmodelle nutzen dieselbe REST-API, wobei in der Dokumentation nur geringfügige Unterschiede zu berücksichtigen sind.

Erweiterung der REST-Ressourcentypen

Die Zahl der Ressourcen, auf die über die Astra Control REST-API zugegriffen werden kann, ist enorm erweitert. Viele der neuen Ressourcen bilden die Grundlage für das On-Premises Astra Control Center-Angebot. Die neuen Ressourcen umfassen: ASUP, Berechtigung, Funktion, Lizenz, Einstellung, Abonnement, Bucket, Cloud, Cluster, gemanagtes Cluster, Back-End-Storage und Storage-Klasse. Siehe "Ressourcen" Finden Sie weitere Informationen.

Zusätzliche Endpunkte unterstützen eine Astra Implementierung

Neben den erweiterten REST-Ressourcen stehen noch mehrere weitere neue API-Endpunkte zur Unterstützung einer Astra Control Implementierung zur Verfügung.

OpenAPI-Unterstützung

Die OpenAPI-Endpunkte bieten Zugriff auf das aktuelle OpenAPI JSON-Dokument und andere zugehörige Ressourcen.

Unterstützung von OpenMetrics

Die OpenMetrics-Endpunkte bieten über die OpenMetrics-Ressource Zugriff auf Kontokennzahlen.

15. April 2021 (21.04)

Diese Version umfasst die folgenden neuen Funktionen und Verbesserungen.

Einführung DER REST API

Die Astra Control REST API ist für den Astra Control Service verfügbar. Das System wurde auf Basis VON REST-Technologien und aktuellen Best Practices erstellt. Die API ist die Grundlage für die Automatisierung Ihrer Astra-Implementierungen und umfasst die folgenden Funktionen und Vorteile.

Ressourcen

Es sind vierzehn REST-Ressourcen verfügbar.

Zugriff auf API-Token

Der Zugriff auf DIE REST-API wird über ein API-Zugriffstoken bereitgestellt, das Sie über die Astra Web-Benutzeroberfläche generieren können. Das API-Token bietet sicheren Zugriff auf die API.

Unterstützung für Sammlungen

Es gibt eine umfangreiche Reihe von Abfrageparametern, die für den Zugriff auf die Ressourcen-Sammlungen verwendet werden können. Einige der unterstützten Vorgänge umfassen Filtern, Sortieren und Paginieren.

Bekannte Probleme

Sie sollten alle bekannten Probleme für die aktuelle Version im Zusammenhang mit der Astra Control REST API überprüfen. Die bekannten Probleme identifizieren Probleme, die die erfolgreiche Verwendung des Produkts verhindern könnten.

Es gibt keine neuen Probleme, die mit der Version 22.04 der Astra Control REST API bekannt sind. Die unten beschriebenen Probleme wurden in vorherigen Versionen entdeckt und gelten noch für die aktuelle Version.

Es werden nicht alle Speichergeräte in einem Back-End-Speicher-Node erkannt

Wenn ein REST-API-Aufruf zum Abrufen der in einem Storage-Node definierten Speichergeräte erfolgt, werden nicht alle Geräte zurückgegeben.

Einführung in die Astra Control REST API

Astra Control Center und Astra Control Service bieten eine gemeinsame REST-API, auf die Sie direkt über eine Programmiersprache oder ein Dienstprogramm wie Curl zugreifen können. Die wichtigsten Highlights und Vorteile der API sind nachfolgend aufgeführt.

Um auf DIE REST-API zuzugreifen, müssen Sie sich zunächst bei der Astra Web-Benutzeroberfläche anmelden und ein API-Token generieren. Sie müssen das Token bei jeder API-Anforderung einschließen.

Basiert auf REST-Technologie

Die Astra Control API wurde mit REST-Technologie und aktuellen Best Practices erstellt. Die Kerntechnologie umfasst HTTP, JSON und RBAC.

Unterstützung der beiden Astra Control Implementierungsmodelle

Astra Control Service wird in der Public Cloud-Umgebung eingesetzt und Astra Control Center ist Ihre lokale Implementierung. Beide Implementierungsmodelle werden über eine REST-API unterstützt.

Klare Zuordnung zwischen REST-Endpunkt-Ressourcen und Objektmodell

Die externen REST-Endpunkte, mit denen auf die Ressourcenzuordnung auf ein konsistentes Objektmodell zugegriffen wird, das vom Astra-Service intern gewartet wird. Das Objektmodell basiert auf er-Modellierung (Entity-Relationship), mit der API-Aktionen und -Antworten klar definiert werden können.

Umfangreiche Reihe von Abfrageparametern

Die REST-API bietet eine umfangreiche Reihe von Abfrageparametern, mit denen Sie auf die Ressourcensammlungen zugreifen können. Einige der unterstützten Vorgänge umfassen Filtern, Sortieren und Paginieren.

Ausrichtung auf die Web-UI von Astra Control

Das Design der Astra Web-Benutzeroberfläche ist auf DIE REST-API abgestimmt, so dass es Konsistenz zwischen den beiden Zugriffspfaden und der Benutzererfahrung gibt.

Robuste Debugging- und Problemermittlung

Die Astra Control REST API bietet eine robuste Debugging- und Problembestimmung, einschließlich Systemereignissen und Benutzerbenachrichtigungen.

Workflow-Prozesse

Sie erhalten eine Reihe von Workflows, die Sie bei der Entwicklung Ihres Automatisierungscodes unterstützen. Die Workflows sind in zwei Kategorien unterteilt: Infrastruktur und Management.

Grundlage für erweiterte Automatisierungstechnologien

Neben dem direkten Zugriff auf DIE REST API können weitere Automatisierungstechnologien verwendet werden, die auf der REST-API basieren.

Teil der Dokumentation der Astra-Familie

Die Dokumentation der Astra Control Automation ist Teil der größeren Dokumentation der Astra-Familie. Siehe "Astra-Dokumentation" Finden Sie weitere Informationen.

Los geht’s

Bevor Sie beginnen

Sie können sich schnell auf den Einstieg in die Astra Control REST API vorbereiten, indem Sie die unten aufgeführten Schritte überprüfen.

Astra-Konto besitzen Anmeldedaten

Sie benötigen Astra-Anmeldeinformationen, um sich bei der Astra Web-Benutzeroberfläche anzumelden und ein API-Token zu generieren. Mit Astra Control Center verwalten Sie diese Anmeldedaten lokal. Mit dem Astra Control Service können Sie über den **Auth0**-Dienst auf die Anmeldeinformationen zugreifen.

Lernen Sie die grundlegenden Kubernetes-Konzepte kennen

Sie sollten mit verschiedenen grundlegenden Kubernetes-Konzepten vertraut sein. Siehe "Grundlegende Kubernetes-Konzepte" Finden Sie weitere Informationen.

ÜBERPRÜFUNG DER REST-Konzepte und der Implementierung

Prüfen Sie die Daten "Core-REST-Implementierung" Für Informationen über REST-Konzepte und die Details zur Entwicklung der Astra Control REST API.

Weitere Informationen

Beachten Sie die zusätzlichen Informationsressourcen, wie in vorgeschlagen "Weitere Ressourcen".

Holen Sie sich ein API-Token

Sie müssen ein Astra API Token erhalten, um die Astra Control REST API zu verwenden.

Einführung

Ein API-Token identifiziert den Anrufer an Astra und muss bei jedem REST-API-Aufruf enthalten sein.

- Sie können über die Astra Web-Benutzeroberfläche ein API-Token generieren.
- Die mit dem Token getragene Benutzeridentität wird vom Benutzer bestimmt, der das Token erstellt.
- Das Token muss in das enthalten sein `Authorization` HTTP-Anfragekopf.
- Ein Token läuft nie ab, nachdem es erstellt wurde.
- Sie können ein Token über die Astra Web-Benutzeroberfläche widerrufen.

Verwandte Informationen

- "Ein API-Token widerrufen"

Erstellen Sie ein Astra API-Token

In den folgenden Schritten wird beschrieben, wie ein Astra API Token erstellt wird.

Bevor Sie beginnen

Sie benötigen die Zugangsdaten für ein Astra-Konto.

Über diese Aufgabe

Diese Aufgabe erzeugt ein API-Token in der Astra-Webschnittstelle. Sie sollten auch die Account-ID abrufen, die auch bei API-Aufrufen benötigt wird.

Schritte

1. Melden Sie sich mit Ihren Anmeldedaten im Astra an.

   Rufen Sie die folgende Website für den Astra Control Service auf: "https://astra.netapp.io"

2. Klicken Sie auf das Figurensymbol oben rechts auf der Seite und wählen Sie **API Access**.
3. Klicken Sie auf **API-Token generieren** auf der Seite und klicken Sie im Popup-Fenster auf **API-Token generieren**.
4. Klicken Sie auf das Symbol, um die Token-Zeichenfolge in die Zwischenablage zu kopieren und im Editor zu speichern.
5. Kopieren Sie die Konto-id, die auf derselben Seite verfügbar ist, und speichern Sie sie.

Nachdem Sie fertig sind

Wenn Sie über Curl oder eine Programmiersprache auf die Astra Control REST API zugreifen, müssen Sie das API-Träger-Token in das HTTP einbeziehen `Authorization` Kopfzeile der Anfrage.

Hallo Welt

Sie können einen einfachen Curl Befehl an Ihrer Workstation CLI ausgeben, um mit der Astra Control REST API zu beginnen und seine Verfügbarkeit zu bestätigen.

Bevor Sie beginnen

Das Dienstprogramm Curl muss auf Ihrer lokalen Workstation verfügbar sein. Sie müssen außerdem über ein API-Token und die zugehörige Account-ID verfügen. Siehe "Holen Sie sich ein API-Token" Finden Sie weitere Informationen.

Beispiel für die Wellung

Der folgende Curl Befehl ruft eine Liste der Astra-Benutzer ab. Geben Sie die entsprechenden <ACCOUNT_ID> und <API_TOKEN> wie angegeben an.

```
curl --location --request GET
'https://astra.netapp.io/accounts/<ACCOUNT_ID>/core/v1/users' --header
'Content-Type: application/json' --header 'Authorization: Bearer
<API_TOKEN>'
```

Beispiel für eine JSON-Ausgabe


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


Die Nutzung der Workflows wird vorbereitet

Vor der Live-Implementierung sollten Sie sich mit der Organisation und dem Format der Astra-Workflows vertraut machen.

Einführung

Ein *Workflow* ist eine Sequenz aus einem oder mehreren Schritten, die zum Erreichen einer bestimmten administrativen Aufgabe oder eines bestimmten Ziels erforderlich sind. Jeder Schritt in einem Astra Control Workflow ist einer der folgenden:

- REST-API-Aufruf (mit Details wie Curl- und JSON-Beispiele)
- Aufruf eines weiteren Astra-Workflows
- Sonstige verwandte Aufgaben (z. B. die Entscheidung für eine erforderliche Designentscheidung)

Workflows umfassen die wichtigsten Schritte und Parameter, die zur Durchführung jeder Aufgabe erforderlich sind. Sie bieten als Ausgangspunkt für die Anpassung Ihrer Automatisierungsumgebung.

Allgemeine Eingabeparameter

Die unten beschriebenen Eingabeparameter sind für alle Curl-Proben, die zur Veranschaulichung eines REST-API-Aufrufs verwendet werden, üblich.

> Da diese Eingabeparameter universell erforderlich sind, werden sie in den einzelnen Arbeitsabläufen nicht weiter beschrieben. Wenn für ein bestimmtes Curl-Beispiel zusätzliche Eingabeparameter verwendet werden, werden sie im Abschnitt **zusätzliche Eingabeparameter** beschrieben.

Pfadparameter

Der bei jedem REST-API-Aufruf verwendete Endpunkt-Pfad umfasst die folgenden Parameter. Siehe auch "URL-Format" Finden Sie weitere Informationen.

Konto-ID

Dies ist der UUIDv4-Wert, der das Astra-Konto identifiziert, auf dem der API-Vorgang ausgeführt wird. Siehe "Holen Sie sich ein API-Token" Weitere Informationen zur Suche nach Ihrer Konto-ID.

Anfragekopfzeilen

Je nach REST-API-Aufruf müssen Sie möglicherweise mehrere Anforderungsheader einbeziehen.

Autorisierung

Für alle API-Aufrufe in den Workflows wird ein API-Token zur Identifizierung des Benutzers benötigt. Sie müssen das Token in das aufnehmen `Authorization` Kopfzeile der Anfrage. Siehe "Holen Sie sich ein API-Token" Weitere Informationen zum Generieren eines API-Tokens finden Sie unter.

Content-Typ

Mit dem HTTP-POST und PUT-Anfragen, bei denen JSON im Anforderungstext enthalten ist, sollten Sie den Medientyp basierend auf der Astra-Ressource deklarieren. Beispielsweise können Sie die Kopfzeile einschließen `Content-Type: application/astra-appSnap+json` Beim Erstellen eines Snapshots für eine verwaltete Anwendung.

Akzeptieren

Sie können den spezifischen Medientyp des Inhalts, den Sie in der Antwort erwarten, basierend auf der Astra-Ressource erklären. Beispielsweise können Sie die Kopfzeile einschließen `Accept: application/astra-appBackup+json` Wenn Sie die Backups für eine gemanagte Applikation auflisten. Zur Vereinfachung akzeptieren die Wellproben in den Workflows jedoch alle Medientypen.

Darstellung von Token und Identifikatoren

Das API-Token und andere ID-Werte, die mit den Curl-Beispielen verwendet werden, sind undurchsichtig und haben keine erkennbare Bedeutung. Um die Lesbarkeit der Proben zu verbessern, werden die tatsächlichen Token- und ID-Werte nicht verwendet. Stattdessen werden kleinere reservierte Schlüsselwörter verwendet, die mehrere Vorteile haben:

- Die Curl- und JSON-Proben sind klarer und leichter zu verstehen.
- Da alle Schlüsselwörter dasselbe Format mit Klammern und Großbuchstaben verwenden, können Sie schnell den Ort und den Inhalt identifizieren, der eingefügt oder extrahiert werden soll.
- Kein Wert geht verloren, da die ursprünglichen Parameter nicht kopiert und bei einer tatsächlichen Implementierung verwendet werden können.

Hier sind einige der in den Curl-Beispielen verwendeten allgemein reservierten Schlüsselwörter. Diese Liste ist nicht vollständig und weitere Schlüsselwörter werden bei Bedarf verwendet. Ihre Bedeutung sollte auf der Grundlage des Kontexts offensichtlich sein.

Stichwort	Typ	Beschreibung
<ACCOUNT_ID>	Pfad	Der UUIDv4-Wert identifiziert das Konto, auf dem der API-Vorgang ausgeführt wird.
<API_TOKEN>	Kopfzeile	Das Inhaberzeichen, das den Anrufer identifiziert und autorisiert.

Stichwort	Typ	Beschreibung
<MANAGED_APP_ID>	Pfad	Der UUIDv4-Wert, der die verwaltete Anwendung für den API-Aufruf identifiziert.

Workflow-Kategorien

Je nach Ihrem Implementierungsmodell stehen Ihnen zwei weit gefassten Kategorien von Astra-Workflows zur Verfügung. Wenn Sie Astra Control Center nutzen, sollten Sie mit den Infrastruktur-Workflows beginnen und anschließend mit den Management-Workflows fortfahren. Mit dem Astra Control Service können Sie in der Regel direkt zu den Management-Workflows wechseln.

> Die Curl-Beispiele in den Workflows verwenden die URL für den Astra Control Service. Sie müssen die URL ändern, wenn Sie das On-Premise Astra Control Center entsprechend Ihrer Umgebung verwenden.

Infrastruktur-Workflows

Diese Workflows befassen sich mit der Astra-Infrastruktur, einschließlich Referenzen, Buckets und Storage-Back-Ends. Sie werden mit dem Astra Control Center benötigt, können aber in den meisten Fällen auch mit dem Astra Control Service verwendet werden. Die Workflows konzentrieren sich auf die Aufgaben, die für die Einrichtung und Wartung eines von Astra gemanagten Clusters erforderlich sind.

Management-Workflows

Diese Workflows können nach einem verwalteten Cluster verwendet werden. Die Workflows konzentrieren sich auf die Applikationssicherung und unterstützen Abläufe wie das Backup, die Wiederherstellung und das Klonen einer gemanagten Applikation.

Grundlegende Kubernetes-Konzepte

Es gibt verschiedene Kubernetes-Konzepte, die für die Verwendung der Astra REST API relevant sind.

Objekte

Die in einer Kubernetes-Umgebung gepflegten Objekte sind persistente Einheiten, die die Konfiguration des Clusters repräsentieren. Diese Objekte beschreiben zusammen den Status des Systems einschließlich des Cluster-Workloads.

Namespaces

Namespaces bieten eine Technik zum Isolation von Ressourcen in einem einzigen Cluster. Diese Organisationsstruktur ist nützlich, wenn die Arten von Arbeit, Nutzer und Ressourcen aufgeteilt werden. Objekte mit einem Umfang „*Namespace“* müssen innerhalb des Namespace eindeutig sein, während Objekte mit einem „*Cluster Scope*“ im gesamten Cluster eindeutig sein müssen.

Etiketten

Labels können den Kubernetes-Objekten zugeordnet werden. Sie beschreiben Attribute mit Schlüsselwert-Paaren und können eine beliebige Organisation auf dem Cluster durchsetzen. Diese können sich für ein Unternehmen nützlich sein, liegen aber nicht in der zentralen Handhabung von Kubernetes.

Core-REST-Implementierung

REST-Web-Services

Representational State Transfer (REST) ist ein Stil für die Erstellung von verteilten Web-Anwendungen. Bei der Anwendung auf das Design einer Web-Services-API, stellt sie eine Reihe von Mainstream-Technologien und Best Practices für die Offenlegung serverbasierter Ressourcen und die Verwaltung ihrer Status. REST bietet eine konsistente Grundlage für die Applikationsentwicklung, doch je nach den jeweiligen Designoptionen können die Details jeder API variieren. Vor dem Einsatz in einer Live-Implementierung sollten Sie sich der Merkmale der Astra Control REST API bewusst sein.

Ressourcen- und Zustandsdarstellung

Ressourcen sind die Grundkomponenten eines webbasierten Systems. Beim Erstellen einer ANWENDUNG FÜR REST-Webservices umfassen die frühen Designaufgaben Folgendes:

- Identifizierung von System- oder serverbasierten Ressourcen

  Jedes System nutzt und verwaltet Ressourcen. Eine Ressource kann eine Datei-, Geschäftstransaktion-, Prozess- oder Verwaltungseinheit sein. Eine der ersten Aufgaben bei der Entwicklung einer auf REST-Webservices basierenden Applikation ist die Identifizierung der Ressourcen.

- Definition von Ressourcenstatus und zugehörigen Statusoperationen

  Die Ressourcen befinden sich immer in einer endlichen Anzahl von Staaten. Die Zustände sowie die damit verbundenen Operationen, die zur Auswirkung der Statusänderungen verwendet werden, müssen klar definiert werden.

URI-Endpunkte

Jede REST-Ressource muss definiert und über ein gut definiertes Adressierungssystem verfügbar gemacht werden. Die Endpunkte, in denen die Ressourcen gefunden und identifiziert werden, verwenden einen einheitlichen Resource Identifier (URI). Der URI bietet ein allgemeines Framework zum Erstellen eines eindeutigen Namens für jede Ressource im Netzwerk. Der Uniform Resource Locator (URL) ist ein URI-Typ, der mit Webservices zur Identifizierung und zum Zugriff von Ressourcen verwendet wird. Ressourcen werden in der Regel in einer hierarchischen Struktur ausgesetzt, die einem Dateiverzeichnis ähnelt.

HTTP-Meldungen

Hypertext Transfer Protocol (HTTP) ist das Protokoll, das vom Webservice-Client und -Server zum Austausch von Anforderungs- und Antwortmeldungen zu den Ressourcen verwendet wird. Im Rahmen der Entwicklung einer Web-Services-Anwendung werden HTTP-Methoden den Ressourcen und entsprechenden Statusmanagement-Aktionen zugeordnet. HTTP ist statusfrei. Um im Rahmen einer Transaktion eine Reihe verwandter Anforderungen und Antworten zuzuordnen, müssen daher zusätzliche Informationen in die HTTP-Header enthalten sein, die mit den Anforderungs- und Antwortdatenströmen verwendet werden.

JSON-Formatierung

Während Informationen auf verschiedene Weise zwischen einem Web-Services-Client und Server strukturiert und übertragen werden können, ist die beliebteste Option JavaScript Object Notation (JSON). JSON ist ein Branchenstandard für die Darstellung einfacher Datenstrukturen im Klartext und wird zur Übertragung von Zustandsdaten zur Beschreibung der Ressourcen verwendet. Die Astra Control REST API verwendet JSON, um die Daten zu formatieren, die im Körper von jeder HTTP-Anfrage und Antwort.

Ressourcen und Sammlungen

DIE Rest-API von Astra Control bietet Zugriff auf Ressourceninstanzen und Ressourcensammlungen.

Konzeptionell ist eine REST **Ressource** ähnlich wie ein **Objekt** wie mit den objektorientierten Programmiersprachen und -Systemen definiert. Manchmal werden diese Begriffe synonym verwendet. Aber im Allgemeinen wird „Ressource“ bevorzugt, wenn sie im Kontext der externen REST API verwendet wird, während „Objekt“ für die entsprechenden zustandsorientierte Instanz Daten verwendet wird, die auf dem Server gespeichert sind.

Eigenschaften der Astra-Ressourcen

Die Astra Control REST API entspricht den Prinzipien des RESTful Designs. Jede Astra-Ressourceninstanz wird auf Basis eines klar definierten Ressourcentyps erstellt. Eine Reihe von Ressourceninstanzen desselben Typs wird als **Sammlung** bezeichnet. Die API-Aufrufe wirken sich auf einzelne Ressourcen oder Ressourcensammlungen aus.

Ressourcentypen

Die in der Astra Control REST API enthaltenen Ressourcentypen weisen folgende Merkmale auf:

- Jeder Ressourcentyp wird mit einem Schema definiert (in der Regel in JSON)
- Jedes Ressourcenschema enthält den Ressourcentyp und die -Version
- Ressourcentypen sind global eindeutig

Ressourceninstanzen

Die über die Astra Control REST-API verfügbaren Ressourceinstanzen weisen folgende Merkmale auf:

- Ressourceninstanzen werden auf Basis eines einzelnen Ressourcentyps erstellt
- Der Ressourcentyp wird mit dem Wert Medientyp angezeigt
- Instanzen bestehen aus statusorientierten Daten, die vom Astra-Service gewartet werden
- Auf jede Instanz kann über eine eindeutige und langlebige URL zugegriffen werden
- In Fällen, in denen eine Ressourceninstanz mehr als eine Darstellung haben kann, können verschiedene Medientypen verwendet werden, um die gewünschte Darstellung anzufordern

Ressourcensammlungen

Die Ressourcensammlungen, die über die ASTRA Control REST-API verfügbar sind, weisen folgende Merkmale auf:

- Der Satz von Ressourceninstanzen eines einzelnen Ressourcentyps wird als Sammlung bezeichnet
- Ressourcensammlungen haben eine einzigartige und langlebige URL

Instanz-IDs

Jeder Ressourceninstanz wird bei der Erstellung eine Kennung zugewiesen. Diese Kennung ist ein 128-Bit UUIDv4-Wert. Die zugewiesenen UUUIDv4-Werte sind global eindeutig und unveränderbar. Nachdem ein API-Aufruf ausgegeben wurde, der eine neue Instanz erstellt, wird eine URL mit der zugehörigen id an den Anrufer in A zurückgegeben `Location` Kopfzeile der HTTP-Antwort. Sie können die Kennung extrahieren und bei nachfolgenden Anrufen verwenden, wenn Sie sich auf die Ressourceninstanz beziehen.

Die Ressourcen-ID ist der primäre Schlüssel, der für Sammlungen verwendet wird.

Gemeinsame Struktur für Astra-Ressourcen

Jede Astra Control-Ressource ist mit einer gemeinsamen Struktur definiert.

Einheitliche Daten

Jede Astra-Ressource enthält die in der folgenden Tabelle aufgeführten Schlüsselwerte.

Taste	Beschreibung
Typ	Ein global eindeutiger Ressourcentyp, der als **Ressourcentyp** bezeichnet wird.
Version	Eine Version-ID, die als **Resource-Version** bezeichnet wird.
id	Ein global eindeutiger Bezeichner, der als **Resource Identifier** bezeichnet wird.
Metadaten	Ein JSON-Objekt mit verschiedenen Informationen, einschließlich Benutzer- und Systemetiketten.

Metadatenobjekt

Das JSON-Metadatenobjekt, das in jeder Astra-Ressource enthalten ist, enthält die in der folgenden Tabelle aufgeführten Schlüsselwerte.

Taste	Beschreibung
Etiketten	JSON-Array mit Client-angegebenen Beschriftungen, die der Ressource zugeordnet sind.
CreationZeitstempel	JSON-Zeichenfolge mit einem Zeitstempel, der angibt, wann die Ressource erstellt wurde.
Änderungszeitstempel	JSON-Zeichenfolge mit einem ISO-8601-formatierten Zeitstempel, der angibt, wann die Ressource zuletzt geändert wurde.
Erstellt von	JSON-Zeichenfolge mit der UUUIDv4-Kennung der Benutzer-id, die die Ressource erstellt hat. Wenn die Ressource von einer internen Systemkomponente erstellt wurde und der Erstellungseinheit keine UUID zugeordnet ist, wird die **Null** UUID verwendet.

Ressourcenstatus

Ausgewählte Ressourcen A `state` Wert, der zur Orchestrierung von Lifecycle-Übergängen und zur Steuerung des Zugriffs eingesetzt wird.

HTTP-Details

Die Astra Control REST API verwendet HTTP und entsprechende Parameter, um auf die Ressourcen und Sammlungen zu reagieren. Einzelheiten zur HTTP-Implementierung finden Sie unten.

API-Transaktionen und das CRUD-Modell

Die Astra Control REST API implementiert ein transaktionsorientiertes Modell mit klar definierten Abläufen und Zustandsübergängen.

API-Transaktion bei Anfrage und Reaktion

Jeder REST-API-Aufruf erfolgt als HTTP-Anfrage an den Astra-Service. Jede Anforderung generiert eine entsprechende Antwort zurück an den Client. Dieses Request-Response-Paar kann als API-Transaktion betrachtet werden.

Unterstützung für CRUD-Betriebsmodell

Auf Grundlage des **CRUD**-Modells kann auf alle über die Astra Control REST API verfügbaren Ressourcen und Sammlungen zugegriffen werden. Es gibt vier Vorgänge, von denen jede einer einzigen HTTP-Methode zugeordnet wird. Dazu gehören:

- Erstellen
- Lesen
- Aktualisierung
- Löschen

Bei einigen der Astra-Ressourcen wird nur ein Teil dieser Vorgänge unterstützt. Sie sollten die überprüfen "API-Referenz" Weitere Informationen zu einem bestimmten API-Aufruf.

HTTP-Methoden

Die von der API unterstützten HTTP-Methoden oder Verben werden in der folgenden Tabelle dargestellt.

Methode	CRUD	Beschreibung
GET	Lesen	Ruft Objekteigenschaften für eine Ressourceninstanz oder -Sammlung ab. Dies wird als **list**-Operation bei Verwendung mit einer Sammlung betrachtet.
POST	Erstellen	Erstellt eine neue Ressourceninstanz basierend auf den Eingabeparametern. Die langfristige URL wird in A zurückgegeben `Location` Kopfzeile der Antwort.
PUT	Aktualisierung	Aktualisiert eine gesamte Ressourceninstanz mit dem mitgelieferten JSON Request Body. Wichtige Werte, die nicht vom Benutzer änderbar sind, bleiben erhalten.
Löschen	Löschen	Löscht eine vorhandene Ressourceninstanz.

Header für Anfragen und Antworten

Die folgende Tabelle fasst die HTTP-Header zusammen, die mit der Astra Control REST API verwendet werden.

Siehe "RFC 7232" Und "RFC 7233" Finden Sie weitere Informationen.

Kopfzeile	Typ	Nutzungshinweise
Akzeptieren	Anfrage	Wenn der Wert „/“ ist oder nicht angegeben wird, `application/json` Wird in der Kopfzeile der Inhaltstyp-Antwort zurückgegeben. Wenn der Wert auf den Astra Resource Media Type gesetzt ist, wird derselbe Medientyp in der Kopfzeile des Inhaltstyps zurückgegeben.
Autorisierung	Anfrage	Träger-Token mit dem API-Schlüssel für den Benutzer.
Inhaltstyp	Antwort	Wird basierend auf dem zurückgegeben `Accept` Kopfzeile der Anfrage.
Etag	Antwort	Im Lieferumfang eines erfolgreichen RFC 7232-Standards enthalten. Der Wert ist eine hexadezimale Darstellung des MD5-Werts für die gesamte JSON-Ressource.
If-Match	Anfrage	Ein Precondition Request Header, wie in Abschnitt 3.1 RFC 7232 beschrieben und unterstützt **PUT** Anforderungen.
Wenn-Geändert-Seit	Anfrage	Ein Precondition Request Header, wie in Abschnitt 3.4 RFC 7232 beschrieben und unterstützt **PUT** Anforderungen.
Wenn-Unmodified-Since	Anfrage	Ein Precondition Request Header, wie in Abschnitt 3.4 RFC 7232 beschrieben und unterstützt **PUT** Anforderungen.
Standort	Antwort	Enthält die vollständige URL der neu erstellten Ressource.

Abfrageparameter

Die folgenden Abfrageparameter stehen zur Verwendung mit Ressourcensammlungen zur Verfügung. Siehe "Arbeiten mit Sammlungen" Finden Sie weitere Informationen.

Abfrageparameter	Beschreibung
Einschließlich	Enthält die Felder, die beim Lesen einer Sammlung zurückgegeben werden sollen.
Filtern	Gibt die Felder an, die für die Rückgabe einer Ressource beim Lesen einer Sammlung übereinstimmen müssen.
Orderby	Bestimmt die Reihenfolge der beim Lesen einer Sammlung zurückgegebenen Ressourcen.
Grenze	Begrenzt die maximale Anzahl an Ressourcen, die beim Lesen einer Sammlung zurückgegeben werden.
überspringen	Legt fest, wie viele Ressourcen beim Lesen einer Sammlung weitergehen und überspringen sollen.
Zählen	Gibt an, ob die Gesamtzahl der Ressourcen im Metadatenobjekt zurückgegeben werden soll.

HTTP-Statuscodes

Im Folgenden werden die HTTP-Statuscodes beschrieben, die von der REST-API von Astra Control verwendet werden.

Die Astra Control REST API nutzt auch den **Problemdetails für HTTP APIs** Standard. Siehe "Diagnose und Support" Finden Sie weitere Informationen.

Codieren	Bedeutung	Beschreibung
200	OK	Zeigt Erfolg für Anrufe an, die keine neue Ressourceninstanz erstellen.
201	Erstellt	Ein Objekt wurde erfolgreich erstellt, und die Kopfzeile für die Standortantwort enthält die eindeutige Kennung für das Objekt.
204	Kein Inhalt	Die Anfrage war erfolgreich, obwohl kein Inhalt zurückgegeben wurde.
400	Schlechte Anfrage	Die Eingabe der Anfrage ist nicht erkannt oder nicht angemessen.
401	Nicht Autorisiert	Der Benutzer ist nicht autorisiert und muss authentieren.
403	Verboten	Der Zugriff wird aufgrund eines Autorisierungsfehlers verweigert.
404	Nicht gefunden	Die Ressource, auf die in diesem Antrag verwiesen wird, ist nicht vorhanden.
409	Konflikt	Der Versuch, ein Objekt zu erstellen, ist fehlgeschlagen, weil das Objekt bereits vorhanden ist.
500	Interner Fehler	Ein allgemeiner interner Fehler ist auf dem Server aufgetreten.
503	Service nicht verfügbar	Der Dienst ist aus irgendeinem Grund nicht bereit, die Anfrage zu bearbeiten.

URL-Format

Die allgemeine Struktur der URL, die für den Zugriff auf eine Ressourceninstanz oder -Sammlung über DIE REST-API verwendet wird, besteht aus mehreren Werten. Diese Struktur spiegelt das zugrunde liegende Objektmodell und das Systemdesign wider.

Konto als Root

Die Wurzel des Ressourcenpfads zu jedem REST-Endpunkt ist das Astra-Konto. Daher beginnen alle Pfade in der URL mit `/account/{account_id}` Wo `account_id` Ist der eindeutige UUUIDv4-Wert für das Konto. Interne Struktur Dies ist ein Design, in dem der gesamte Ressourcenzugriff auf einem bestimmten Konto basiert.

Kategorie der Endpoint-Ressourcen

Die Astra-Ressourcenendpunkte lassen sich in drei verschiedene Kategorien einteilen:

- Kern (`/core`)
- Gemanagte Applikation (`/k8s`)
- Topologie (`/topology`)

Siehe "Ressourcen" Finden Sie weitere Informationen.

Kategorienversion

Jede der drei Ressourcenkategorien verfügt über eine globale Version, die die Version der Ressourcen steuert, auf die zugegriffen wird. Nach Konventionen und Definition zu einer neuen Hauptversion einer Ressourcenkategorie wechseln (z. B. von) `/v1` Bis `/v2`) Wird Bruchänderungen in der API einführen.

Ressourceinstanz oder -Sammlung

Eine Kombination von Ressourcentypen und Identifikatoren kann im Pfad verwendet werden, basierend darauf, ob auf eine Ressourceninstanz oder -Sammlung zugegriffen wird.

Beispiel

- Ressourcenpfad

  Basierend auf der oben dargestellten Struktur ist ein typischer Pfad zu einem Endpunkt: `/accounts/{account_id}/core/v1/users`.

- Vollständige URL

  Die vollständige URL für den entsprechenden Endpunkt lautet: `https://astra.netapp.io/accounts/{account_id}/core/v1/users`.

Ressourcen und Endpunkte

Sie können die Ressourcen, die über die Astra Control REST-API bereitgestellt werden, nutzen, um eine Astra Implementierung zu automatisieren. Jede Ressource ist über einen oder mehrere Endpunkte zugänglich. Die folgenden Informationen bieten eine Einführung in DIE REST-Ressourcen, die Sie im Rahmen einer Automatisierungsimplementierung nutzen können.

Das Format des Pfads und der vollständigen URL für den Zugriff auf die Astra Control-Ressourcen basiert auf mehreren Werten. Siehe "URL-Format" Finden Sie weitere Informationen. Siehe auch "API-Referenz" Weitere Informationen zur Verwendung der Astra-Ressourcen und -Endpunkte.

Zusammenfassung der Astra Control REST-Ressourcen

Die primären Ressourcenendpunkte in der Astra Control REST API sind in drei Kategorien unterteilt. Auf jede Ressource kann mit allen CRUD-Vorgängen (Erstellen, Lesen, Aktualisieren, Löschen) zugegriffen werden, sofern nicht anders angegeben.

Die Spalte **Release** zeigt den Astra-Release an, als die Ressource zum ersten Mal eingeführt wurde. Dieses Feld ist für Ressourcen verfügbar, die mit der aktuellen Version neu hinzugefügt wurden.

Kernressourcen

Die Kernressourcenendpunkte bieten die grundlegenden Services, die zum Aufbau und zur Wartung der Astra-Laufzeitumgebung erforderlich sind.

Ressource	Freigabe	Beschreibung
Konto	21.12	Mithilfe der Account-Ressourcen können Sie die isolierten Mandanten innerhalb der mandantenfähigen Astra Control Implementierungsumgebung managen.
ASUP	21.08	Die ASUP Ressourcen stellen die AutoSupport Bundles dar, die an den NetApp Support weitergeleitet werden.
Anmeldedaten	21.04	Die Ressourcen für Zugangsdaten enthalten sicherheitsbezogene Informationen, die mit Astra-Benutzern, Clustern, Buckets und Storage-Back-Ends verwendet werden können.
Berechtigung	21.08	Die Berechtigungsressourcen stellen die Funktionen und Kapazitäten dar, die für ein Konto auf Basis der aktiven Lizenzen und Abonnements verfügbar sind.
Ereignis	21.04	Die Event-Ressourcen repräsentieren alle Ereignisse, die im System auftreten, einschließlich der Untergruppe, die als Benachrichtigungen klassifiziert ist.
Execution Hook	21.12	Die Hook-Ressourcen für die Ausführung stellen benutzerdefinierte Skripts dar, die Sie entweder vor oder nach einem Snapshot einer verwalteten App ausführen können.

Ressource	Freigabe	Beschreibung
Merkmal	21.08	Die Funktionsressourcen stellen ausgewählte Astra-Funktionen dar, die Sie abfragen können, um festzustellen, ob diese im System aktiviert oder deaktiviert sind. Der Zugriff ist auf schreibgeschützt beschränkt.
Hook-Quelle	21.12	Die Hakenquellenressourcen stellen den aktuellen Quellcode dar, der mit einem Testsuite verwendet wird. Die Trennung des Quellcodes von der Ausführungskontrolle hat mehrere Vorteile, wie z. B. die Freigabe der Skripte.
Lizenz	21.08	Die Lizenzressourcen stellen die für ein Astra-Konto verfügbaren Lizenzen dar.
Benachrichtigung	21.04	Die Benachrichtigungsressourcen sind Astra-Ereignisse mit einem Benachrichtigungsziel. Der Zugriff erfolgt auf Benutzerbasis.
Paket	**22.04**	Die Paketressourcen ermöglichen die Registrierung und den Zugriff auf Paketdefinitionen. Softwarepakete bestehen aus verschiedenen Komponenten, einschließlich Dateien, Bildern und anderen Artefakten.
Rollenbindung	21.04	Die Role Binding Ressourcen stellen die Beziehungen zwischen bestimmten Paaren von Benutzern und Konten dar. Zusätzlich zur Verknüpfung zwischen den beiden wird für jede über eine bestimmte Rolle ein Satz von Berechtigungen festgelegt.
Einstellung	21.08	Die Einstellungsressourcen stellen eine Sammlung von Schlüsselwert-Paaren dar, die ein Feature für ein bestimmtes Astra-Konto beschreiben.
Abonnement	21.08	Die Abonnementressourcen stellen die aktiven Abonnements für ein Astra-Konto dar.
Token	21.04	Die Token-Ressourcen stellen die Token dar, die für den programmatischen Zugriff auf die Astra Control REST API verfügbar sind.
Ungelesene Benachrichtigung	21.04	Die nicht gelesenen Benachrichtigungsressourcen stellen Benachrichtigungen dar, die einem bestimmten Benutzer zugewiesen, aber noch nicht gelesen wurden.
Upgrade	**22.04**	Die Upgrade-Ressourcen bieten Zugriff auf Softwarekomponenten und können Upgrades initiieren.
Benutzer	21.04	Die Benutzerressourcen sind Astra-Benutzer, die auf das System basierend auf ihrer definierten Rolle zugreifen können.

Gemanagte Applikationsressourcen

Die Endpunkte der gemanagten Applikationsressourcen bieten Zugriff auf die gemanagten Kubernetes-Applikationen.

Ressource	Freigabe	Beschreibung
Anwendungsressource	21.04	Die Anwendungsressourcen stellen interne Sammlungen von staatlichen Informationen dar, die für das Management der Astra-Anwendungen erforderlich sind.
Applikations-Backup	21.04	Die Backup-Ressourcen der Applikation stellen Backups der gemanagten Applikationen dar.

Ressource	Freigabe	Beschreibung
Anwendungs-Snapshot	21.04	Die Snapshot-Ressourcen der Anwendung stellen Snapshots der verwalteten Anwendungen dar.
Überschreiben des Testablaufanhängehaken	21.12	Über die Ressourcen zum Überschreiben der Execution Hooks können Sie die vorab geladenen NetApp Standard-Testausführungshaken für bestimmte Applikationen nach Bedarf deaktivieren.
Gemanagte Applikation	21.04	Die gemanagten App-Ressourcen stellen Kubernetes-Applikationen dar, die von Astra gemanagt werden.
Zeitplan	21.04	Die Zeitplanressourcen sind Datensicherungsvorgänge, die im Rahmen einer Datenschutzrichtlinie für die gemanagten Applikationen geplant sind.

Topologieressourcen

Die Endpunkte der Topologieressourcen bieten Zugriff auf nicht verwaltete Applikationen und Storage-Ressourcen.

Ressource	Freigabe	Beschreibung
App.	21.04	Die App-Ressourcen umfassen alle Kubernetes-Applikationen, auch die, die nicht von Astra gemanagt werden.
Eimer	21.08	Die Bucket-Ressourcen sind die S3-Cloud-Buckets, die für die Speicherung von Backups der vom Astra gemanagten Applikationen verwendet werden.
Cloud	21.08	Die Cloud-Ressourcen stellen Clouds dar, mit denen Astra-Clients verbunden werden können, um Cluster und Applikationen zu managen.
Cluster	21.08	Die Cluster-Ressourcen stellen die Kubernetes-Cluster dar, die nicht von Kubernetes gemanagt werden.
Cluster-Node	21.12	Die Cluster-Node-Ressourcen bieten eine zusätzliche Auflösung, durch die Sie auf die einzelnen Nodes innerhalb eines Kubernetes-Clusters zugreifen können.
Verwalteter Cluster	21.08	Die gemanagten Cluster-Ressourcen stellen die Kubernetes-Cluster dar, die derzeit von Kubernetes gemanagt werden.
Gemanagtes Storage-Back-End	21.12	Die gemanagten Storage-Backend-Ressourcen ermöglichen Ihnen den Zugriff auf abstrahierte Darstellungen der Back-End-Storage-Anbieter. Diese Storage-Back-Ends können von den gemanagten Clustern und Applikationen verwendet werden.
Namespace	21.12	Die Namespace-Ressourcen bieten Zugriff auf die innerhalb eines Kubernetes-Clusters verwendeten Namespaces.
Storage-Back-End	21.08	Die Storage-Back-End-Ressourcen stellen Anbieter von Storage-Services dar, die von den von Astra gemanagten Clustern und Applikationen verwendet werden können.
Storage-Klasse	21.08	Ressourcen der Storage-Klasse stellen unterschiedliche Storage-Klassen oder -Typen dar, die für ein bestimmtes gemanagtes Cluster erkannt und verfügbar sind.
Datenmenge	21.04	Die Volume-Ressourcen stellen die Kubernetes Storage Volumes dar, die mit den gemanagten Applikationen verknüpft sind.

Neue Endpunkte mit der aktuellen Version

Die folgenden REST-Endpunkte wurden mit der aktuellen Version 22.04 Astra Control hinzugefügt. Außerdem wurden die Versionen verschiedener vorhandener Ressourcen aktualisiert.

- /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/Clouds/{Cloud_id}/Cluster/{Cluster_id}/clusterNodes
- /Accounts/{Account_id}/Topology/v1/Clouds/{Cloud_id}/Cluster/{Cluster_id}/clusterNodes/{clusterNode_id}
- /Accounts/{Account_id}/Topology/v1/manageCluster/{managedCluster_id}/Apps/{App_id}/appAssets
- /Accounts/{Account_id}/Topology/v1/manageCluster/{managedCluster_id}/Apps/{App_id}/appAssets/{appAsset_id}
- /Accounts/{Account_id}/Topology/v1/manageCluster/{managedCluster_id}/clusterNodes
- /Accounts/{Account_id}/Topology/v1/manageCluster/{managedCluster_id}/clusterNodes/{clusterNode_id}

Zusätzliche Ressourcen und Endpunkte

Zur Unterstützung einer Astra-Implementierung stehen mehrere zusätzliche Ressourcen und Endpunkte zur Verfügung.

Diese Ressourcen und Endpunkte sind derzeit nicht in der Astra Control REST API-Referenzdokumentation enthalten.

OpenAPI

Die OpenAPI-Endpunkte bieten Zugriff auf das aktuelle OpenAPI JSON-Dokument und andere zugehörige Ressourcen.

OpenMetrics

Die OpenMetrics-Endpunkte bieten über die OpenMetrics-Ressource Zugriff auf die Kontokennzahlen. Support ist mit dem Astra Control Center Implementierungsmodell verfügbar.

Weitere Nutzungsüberlegungen

RBAC-Sicherheit

Die Astra REST API unterstützt die rollenbasierte Zugriffssteuerung (RBAC) zur Beschränkung des Zugriffs auf Systemfunktionen.

Astra Rollen

Jeder Astra-Benutzer wird einer einzigen Rolle zugewiesen, die die Aktionen bestimmt, die durchgeführt werden können. Die Rollen sind in einer Hierarchie angeordnet, wie in der folgenden Tabelle beschrieben.

Rolle	Beschreibung
Eigentümer	Hat alle Berechtigungen der Admin-Rolle und kann auch Astra-Konten löschen.
Admin	Verfügt über alle Berechtigungen der Mitgliedsrolle und kann Benutzer auch dazu einladen, einem Konto beizutreten.
Mitglied	Kunden können ihre Astra-Applikations- und Computing-Ressourcen vollständig managen.
Prüfer	Beschränkt auf die Anzeige von Ressourcen.

Erweiterte RBAC mit Namespace-Granularität

Diese Funktion wurde mit Version 22.04 des Astra REST API eingeführt.

Wenn eine Rollenbindung für einen bestimmten Benutzer festgelegt wird, kann eine Einschränkung angewendet werden, um die Namespaces zu begrenzen, auf die der Benutzer Zugriff hat. Diese Bedingung kann auf verschiedene Weise definiert werden, wie in der nachstehenden Tabelle beschrieben. Siehe Parameter `roleContraints` In der Role Binding API für weitere Informationen.

Namespaces	Beschreibung
Alle	Der Benutzer kann über den Platzhalterparameter „*“ auf alle Namespaces zugreifen. Dies ist der Standardwert, um die Abwärtskompatibilität beizubehalten.
Keine	Die Bedingungsliste wird angegeben, obwohl sie leer ist. Dies bedeutet, dass der Benutzer keinen Zugriff auf einen Namespace hat.
Namespace-Liste	Die UUID eines Namespace enthält, die den Benutzer auf den Single Namespace beschränkt. Eine kommagetrennte Liste kann auch verwendet werden, um den Zugriff auf mehrere Namespaces zu ermöglichen.
Etikett	Ein Etikett wird angegeben und der Zugriff ist allen übereinstimmenden Namespaces erlaubt.

Arbeit mit Sammlungen

Die Astra Control REST API bietet verschiedene Möglichkeiten, über die definierten Abfrageparameter auf Ressourcensammlungen zuzugreifen.

Wählen Sie Werte aus

Sie können angeben, welche Schlüsselwertpaare für jede Ressourceninstanz mit dem zurückgegeben werden sollen `include` Parameter. Alle Fälle werden im Antwortkörper zurückgegeben.

Filtern

Mithilfe der Filterung von Sammlungsressourcen kann ein API-Benutzer Bedingungen festlegen, die bestimmen, ob eine Ressource im Antwortkörper zurückgegeben wird. Der `filter` Parameter wird verwendet, um die Filterbedingung anzuzeigen.

Sortieren

Die Sammelressource-Sortierung ermöglicht einem API-Benutzer, die Reihenfolge anzugeben, in der Ressourcen im Antwortkörper zurückgegeben werden. Der `orderBy` Parameter wird verwendet, um die Filterbedingung anzuzeigen.

Paginierung

Sie können Paginierung erzwingen, indem Sie die Anzahl der Ressourceninstanzen beschränken, die für eine Anforderung über die zurückgegeben werden `limit` Parameter.

Zählen

Wenn Sie den Booleschen Parameter angeben `count` Auf einstellen `true`, Die Anzahl der Ressourcen im zurückgegebenen Array für eine bestimmte Antwort ist im Abschnitt Metadaten angegeben.

Diagnose und Support

Mit der Astra Control REST API stehen verschiedene Supportfunktionen zur Verfügung, die für Diagnose und Debugging genutzt werden können.

API-Ressourcen

Verschiedene Astra-Funktionen sind über API-Ressourcen zugänglich und bieten diagnostische Informationen und Support.

Typ	Beschreibung
Ereignis	Systemaktivitäten, die im Rahmen der Astra-Verarbeitung erfasst werden.
Benachrichtigung	Eine Untergruppe der Ereignisse, die als wichtig genug betrachtet werden, um dem Benutzer präsentiert zu werden.
Ungelesene Benachrichtigung	Die Benachrichtigungen, die noch vom Benutzer gelesen oder abgerufen werden müssen.

Ein API-Token widerrufen

Sie können ein API-Token an der Astra-Webschnittstelle widerrufen, wenn es nicht mehr benötigt wird.

Bevor Sie beginnen

Sie benötigen ein Astra-Konto. Sie sollten auch die Token identifizieren, die Sie widerrufen möchten.

Über diese Aufgabe

Nachdem ein Token entzogen wurde, ist es sofort und dauerhaft unbrauchbar.

Schritte

1. Melden Sie sich mit Ihren Anmeldedaten im Astra an.

   Rufen Sie die folgende Website für den Astra Control Service auf: "https://astra.netapp.io"

2. Klicken Sie auf das Figurensymbol oben rechts auf der Seite und wählen Sie **API Access**.
3. Wählen Sie das Token oder die Token aus, die Sie widerrufen möchten.
4. Klicken Sie im Dropdown-Feld **Aktionen** auf **Token aufheben**.

Infrastruktur-Workflows

Bevor Sie beginnen

Mithilfe dieser Workflows können Sie die Infrastruktur erstellen und pflegen, die mit dem Astra Control Center Implementierungsmodell verwendet wird. In den meisten Fällen können die Workflows auch mit dem Astra Control Service genutzt werden.

Diese Workflows können jederzeit von NetApp erweitert und ergänzt werden, sodass Sie sie in regelmäßigen Abständen prüfen sollten.

Allgemeine Vorbereitung

Bevor Sie einen Astra-Workflow verwenden, sollten Sie unbedingt lesen "Die Nutzung der Workflows wird vorbereitet".

Workflow-Kategorien

Die Infrastruktur-Workflows sind in verschiedene Kategorien unterteilt, um den gewünschten Workflow leichter finden zu können.

Kategorie	Beschreibung
Identität und Zugriff	Mit diesen Workflows können Sie die Identität und den Zugriff auf Astra verwalten. Zu den Ressourcen zählen Benutzer, Anmeldedaten und Token.
Buckets	Sie können diese Workflows zum Erstellen und Managen der S3-Buckets verwenden, die zum Speichern von Backups verwendet werden.
Storage	Durch diese Workflows können Sie Storage-Back-Ends und -Volumes hinzufügen und verwalten.
Cluster	Sie können Managed Kubernetes Cluster hinzufügen, damit Sie die enthaltenen Applikationen schützen und unterstützen können.

Identität und Zugriff

Benutzer auflisten

Sie können die Benutzer auflisten, die für ein bestimmtes Astra-Konto definiert sind.

1. Listen Sie die Benutzer auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Core/v1/users

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Einschließlich	Abfrage	Nein	Wählen Sie optional die Werte aus, die in der Antwort zurückgegeben werden sollen.

Curl-Beispiel: Alle Daten für alle Benutzer zurückgeben

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

Curl Beispiel: Gibt den vor-, Nachnamen und die id für alle Benutzer zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe


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

Buckets

Buckets auflisten

Sie können die S3-Buckets für ein bestimmtes Astra-Konto auflisten.

1. Listen Sie die Eimer auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Topology/v1/Buckets

Curl-Beispiel: Gibt alle Daten für alle Buckets zurück

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

Storage

Auflisten von Storage-Back-Ends

Sie können die verfügbaren Storage-Back-Ends auflisten.

1. Listen Sie die Eimer auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Topology/v1/storageBackends

Curl-Beispiel: Gibt alle Daten für alle Storage-Back-Ends zurück

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

Beispiel für eine JSON-Ausgabe

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


Cluster

Auflistung gemanagter Cluster

Sie können die Kubernetes-Cluster auflisten, die derzeit vom Astra gemanagt werden.

1. Listen Sie die Cluster auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Topology/v1/manageClusters

Curl-Beispiel: Gibt alle Daten für alle Cluster zurück

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

Management-Workflows

Bevor Sie beginnen

Diese Workflows können als Teil der Verwaltung der Applikationen in einem von Astra gemanagten Cluster verwendet werden.

Diese Workflows können jederzeit von NetApp erweitert und ergänzt werden, sodass Sie sie in regelmäßigen Abständen prüfen sollten.

Allgemeine Vorbereitung

Bevor Sie einen Astra-Workflow verwenden, sollten Sie unbedingt lesen "Die Nutzung der Workflows wird vorbereitet".

Workflow-Kategorien

Die Management-Workflows sind in verschiedene Kategorien unterteilt, um die Suche nach den gewünschten zu erleichtern.

Kategorie	Beschreibung
Applikationskontrolle	Mithilfe dieser Workflows können Sie verwaltete und nicht gemanagte Applikationen steuern. Sie können die Apps auflisten sowie eine verwaltete App erstellen und entfernen.
Applikationssicherung	Mithilfe dieser Workflows können gemanagte Applikationen durch Snapshots und Backups gesichert werden.
Klonen und Wiederherstellen von Applikationen	In diesen Workflows wird beschrieben, wie Sie gemanagte Applikationen klonen und wiederherstellen.
Unterstützung	Es stehen mehrere Workflows zur Verfügung, um Ihre Applikationen zu debuggen und zu unterstützen sowie die allgemeine Kubernetes-Umgebung.

Weitere Überlegungen

Bei der Verwendung der Management-Workflows müssen einige zusätzliche Aspekte berücksichtigt werden.

Klonen einer Applikation

Beim Klonen einer Applikation müssen einige Aspekte berücksichtigt werden. Die unten beschriebenen Parameter sind Teil des JSON-Eingangs.

Quell-Cluster-ID

Der Wert von `sourceClusterID` Identifiziert immer das Cluster, auf dem die ursprüngliche App installiert ist.

Cluster-ID

Der Wert von `clusterID` Identifiziert das Cluster, auf dem die neue App installiert werden soll.

- Beim Klonen innerhalb desselben Clusters `clusterID` Und `sourceClusterID` Gleicher Wert.
- Beim Klonen über Cluster unterscheiden sich die beiden Werte und `clusterID` Sollte die ID des Ziel-Clusters sein.

Namespaces

Der `namespace` Der Wert muss sich von der ursprünglichen Quell-App unterscheiden. Außerdem kann der Namespace für den Klon nicht vorhanden sein und Astra wird ihn erstellen.

Backups und Snapshots

Optional können Sie eine Applikation aus einem vorhandenen Backup oder Snapshot mit der klonen `backupID` Oder `snapshotID` Parameter. Wenn Sie keine Backup oder Momentaufnahme zur Verfügung stellen, wird Astra zuerst eine Sicherung der Anwendung erstellen und dann aus dem Backup klonen.

Wiederherstellen einer Anwendung

Folgende Punkte sind bei der Wiederherstellung einer Applikation zu beachten.

- Das Wiederherstellen einer Applikation ähnelt dem Klonvorgang.
- Beim Wiederherstellen einer Anwendung müssen Sie entweder ein Backup oder einen Snapshot bereitstellen.

Applikationskontrolle

Listen Sie die nicht verwalteten Apps auf

Sie können die Applikationen auflisten, die aktuell nicht vom Astra gemanagt werden. Sie können dies als Teil der Auswahl einer zu verwaltenden App tun.

DER REST-Endpunkt, der in diesen Workflows verwendet wird, gibt standardmäßig alle Astra-Anwendungen zurück. Sie können das verwenden `filter` Abfrage-Parameter auf dem API-Aufruf, um nur die nicht verwalteten Apps anzufordern. Alternativ können Sie den Filterparameter weglassen, um alle Apps zurückzugeben, und dann die prüfen `managedState` Feld in der Ausgabe, um zu bestimmen, welche Apps in der enthalten sind `unmanaged` Bundesland.

Auflisten Sie nur die Applikationen mit managed Zustand gleich unverwaltet

In diesem Workflow wird das verwendet `filter` Abfrage-Parameter, um nur die nicht verwalteten Apps zurückzugeben.

1. Listen Sie die nicht verwalteten Anwendungen auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Topology/v1/Apps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Filtern	Abfrage	Nein	Verwenden Sie einen Filter, um festzulegen, welche Apps zurückgegeben werden sollen.
Einschließlich	Abfrage	Nein	Wählen Sie optional die Werte aus, die in der Antwort zurückgegeben werden sollen.

Curl Beispiel: Geben Sie den Namen, id und managedState für die nicht verwalteten Apps zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe

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


Listen Sie alle Apps auf, und wählen Sie die nicht verwalteten Apps aus

Dieser Workflow gibt alle Apps zurück. Sie müssen die Ausgabe überprüfen, um festzustellen, welche nicht verwaltet werden.

1. Listen Sie alle Anwendungen auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Topology/v1/Apps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Einschließlich	Abfrage	Nein	Wählen Sie optional die Werte aus, die in der Antwort zurückgegeben werden sollen.

Curl Beispiel: Gibt alle Daten für alle Apps zurück

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

Curl Beispiel: Geben Sie den Namen, id und managedState für alle Apps zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe

```
{
  "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. Wählen Sie die nicht verwalteten Anwendungen

Überprüfen Sie die Ausgabe des API-Anrufs, und wählen Sie die Apps manuell mit aus `managedState` Gleich `unmanaged`.

Listen Sie die verwalteten Apps auf

Sie können die Applikationen auflisten, die aktuell vom Astra verwaltet werden. Dies könnten Sie tun, um die Snapshots oder Backups für eine bestimmte Anwendung zu finden.

1. Listen Sie die Anwendungen auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/k8s/v1/manageApps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Einschließlich	Abfrage	Nein	Wählen Sie optional die Werte aus, die in der Antwort zurückgegeben werden sollen.

Curl Beispiel: Gibt alle Daten für alle Apps zurück

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

Curl-Beispiel: Gibt den Namen, die id und den Status aller Apps zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe

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


Lassen Sie eine gemanagte App herunter

Sie können alle Ressourcenvariablen abrufen, die eine einzelne verwaltete Anwendung beschreiben.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, die Sie abrufen möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Holen Sie sich die Anwendung

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Accounts/{Account_id}/k8s/v1/manageApps/{managedApp_id}

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	ID-Wert der abzurufenden verwalteten Anwendung.

Curl Beispiel: Alle Daten für die Anwendung zurückgeben

```
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>'
```

Eine App verwalten

Sie können eine gemanagte Anwendung auf Basis einer bereits bekannten Astra-Anwendung erstellen. Wenn eine Applikation gemanagt wird, können Sie die Daten durch regelmäßige Backups und Snapshots sichern.

Bevor Sie beginnen

Sie müssen über die ID der erkannten App verfügen, die Sie verwalten möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die nicht verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Verwalten Sie die Anwendung

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Account/{AccountID}/k8s/v1/manageApps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
JSON	Text	Ja.	Stellt die Parameter bereit, die zur Identifizierung der zu verwaltenden Anwendung erforderlich sind. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

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

Curl Beispiel: Eine App verwalten

```
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
```

Verwaltung einer Anwendung aufheben

Sie können eine verwaltete Anwendung entfernen, wenn sie nicht mehr benötigt wird. Durch Entfernen einer verwalteten Anwendung werden auch die zugeordneten Zeitpläne gelöscht.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, die Sie verwalten möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

Backups und Snapshots der Applikation werden nicht automatisch entfernt, wenn sie gelöscht wird. Wenn Sie die Backups und Snapshots nicht mehr benötigen, sollten Sie sie löschen, bevor Sie die Anwendung entfernen.

1. Die App wurde nicht verwaltet

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
Löschen	/Accounts/{Account_id}/k8s/v1/manageApps/{managedApp_id}

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Identifiziert die zu entfernende verwaltete Anwendung.

Curl Beispiel: Eine verwaltete App entfernen

```
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>'
```

App-Schutz

Listen Sie die Snapshots auf

Sie können die Snapshots auflisten, die für eine bestimmte verwaltete Anwendung erstellt wurden.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, für die Sie die Snapshots auflisten möchten. Bei Bedarf

können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Listen Sie die Snapshots auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Accounts/{Account_id}/k8s/v1/managedaApps/{managedApp_id}/appSnaps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Identifiziert die verwaltete Anwendung, die die aufgeführten Snapshots besitzt.
Zählen	Abfrage	Nein	Wenn `count=true` Die Anzahl der Snapshots wird im Metadatenabschnitt der Antwort berücksichtigt.

Curl Beispiel: Gibt alle Schnappschüsse für die App zurück

```
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 Beispiel: Gibt alle Snapshots für die App und die Zählung zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe

```
{
  "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
  }
}
```


Listen Sie die Backups auf

Sie können die Backups auflisten, die für eine bestimmte verwaltete Anwendung erstellt wurden.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, für die Sie die Backups auflisten möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Listen Sie die Backups auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Accounts/{Account_id}/k8s/v1/managedaApps/{managedApp_id}/appBackups

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Gibt die verwaltete Anwendung an, die die aufgeführten Backups besitzt.

Curl Beispiel: Alle Backups für die App zurückgeben

```
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>'
```

Beispiel für eine JSON-Ausgabe


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


Erstellen Sie einen Snapshot für eine verwaltete Anwendung

Sie können einen Snapshot für eine bestimmte verwaltete Anwendung erstellen.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, für die Sie einen Snapshot erstellen möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Erstellen Sie einen Snapshot

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Accounts/{Account_id}/k8s/v1/managedaApps/{managedApp_id}/appSnaps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Gibt die verwaltete Anwendung an, in der der Snapshot erstellt werden soll.
JSON	Text	Ja.	Stellt die Parameter für den Snapshot bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

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

Curl Beispiel: Erstellen Sie einen Snapshot für die App

```
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
```

Backup für eine gemanagte Applikation erstellen

Sie können ein Backup für eine bestimmte gemanagte Applikation erstellen. Sie können das Backup zum Wiederherstellen oder Klonen der App verwenden.

Bevor Sie beginnen

Sie müssen über die ID der verwalteten App verfügen, für die Sie ein Backup erstellen möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.

1. Erstellen Sie ein Backup

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Accounts/{Account_id}/k8s/v1/managedaApps/{managedApp_id}/appBackups

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Gibt die gemanagte Applikation an, in der das Backup erstellt wird.
JSON	Text	Ja.	Stellt die Parameter für die Sicherung bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

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

Curl Beispiel: Erstellen Sie ein Backup für die App

```
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
```

Löschen Sie einen Snapshot

Sie können einen Snapshot löschen, der einer verwalteten Anwendung zugeordnet ist.

Bevor Sie beginnen

Sie müssen Folgendes haben:

- ID der verwalteten App, die den Snapshot besitzt. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.
- ID des Snapshots, den Sie löschen möchten. Bei Bedarf können Sie den Workflow verwenden "Listen Sie

die Snapshots auf" Um den Snapshot zu finden.

1. Löschen Sie den Snapshot

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
Löschen	/Accounts/{Account_id}/k8s/v1/managedApps/{managedApp_id}/appSnaps/{appSnap_id}

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Identifiziert die verwaltete Anwendung, die den Snapshot besitzt.
snapshot-id	Pfad	Ja.	Identifiziert den zu löschenden Snapshot.

Curl Beispiel: Löschen Sie einen einzelnen Snapshot für die App

```
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>'
```

Löschen Sie ein Backup

Sie können ein Backup löschen, das einer verwalteten Anwendung zugeordnet ist.

Bevor Sie beginnen

Sie müssen Folgendes haben:

- ID der gemanagten Applikation, für die das Backup zuständig ist Bei Bedarf können Sie den Workflow verwenden "Listen Sie die verwalteten Apps auf" Zum Auffinden der Anwendung.
- ID des zu löschenden Backups. Bei Bedarf können Sie den Workflow verwenden "Listen Sie die Backups auf" Um den Snapshot zu finden.

1. Löschen Sie die Sicherung

Führen Sie den folgenden REST-API-Aufruf aus.

Sie können das Löschen einer fehlgeschlagenen Sicherung wie unten beschrieben mit der optionalen Anforderungs-Kopfzeile erzwingen.

HTTP-Methode	Pfad
Löschen	/Accounts/{Account_id}/k8s/v1/managedaApps/{managedApp_id}/appBackups/{appBackup_id}

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Verwaltete App-id	Pfad	Ja.	Gibt die verwaltete Anwendung an, die das Backup besitzt.
Backup-id	Pfad	Ja.	Identifiziert das zu löschende Backup.
Löschen erzwingen	Kopfzeile	Nein	Wird verwendet, um das Löschen eines fehlgeschlagenen Backups zu erzwingen.

Curl Beispiel: Löschen Sie ein einzelnes Backup für die App

```
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 Beispiel: Löschen Sie eine einzelne Sicherung für die App mit der Force-Option

```
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'
```

Klonen und Wiederherstellen einer Applikation

Gemanagte Applikation klonen

Sie können eine neue Applikation erstellen, indem Sie eine bereits gemanagte Applikation klonen.

Bevor Sie beginnen

Beachten Sie Folgendes zu diesem Workflow:

- Ein Anwendungsbackup oder -Snapshot wird nicht verwendet
- Der Klonvorgang wird im selben Cluster durchgeführt

Zum Klonen einer App auf einem anderen Cluster müssen Sie den aktualisieren `clusterId` Parameter in den JSON-Input, wie es für Ihre Umgebung geeignet ist.

1. Wählen Sie die verwaltete App zum Klonen aus

Führen Sie den Workflow aus "Listen Sie die verwalteten Apps auf" Und wählen Sie die Anwendung aus, die Sie klonen möchten. Für DEN REST-Aufruf, der zum Klonen der App verwendet wird, sind mehrere Ressourcenwerte erforderlich.

2. Die App klonen

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Account/{AccountID}/k8s/v1/manageApps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
JSON	Text	Ja.	Stellt die Parameter für die geklonte App bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

```
{
  "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 Beispiel: Klonen einer App

```
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
```

Klonen einer verwalteten Anwendung aus einem Snapshot

Sie können eine neue Anwendung erstellen, indem Sie sie über einen App-Snapshot klonen.

Bevor Sie beginnen

Beachten Sie Folgendes zu diesem Workflow:

- Ein App-Snapshot wird verwendet
- Der Klonvorgang wird im selben Cluster durchgeführt

> Zum Klonen einer App auf einem anderen Cluster müssen Sie den aktualisieren `clusterId` Parameter in den JSON-Input, wie es für Ihre Umgebung geeignet ist.

1. Wählen Sie die verwaltete App zum Klonen aus

Führen Sie den Workflow aus "Listen Sie die verwalteten Apps auf" Und wählen Sie die Anwendung aus, die Sie klonen möchten. Für DEN REST-Aufruf, der zum Klonen der App verwendet wird, sind mehrere Ressourcenwerte erforderlich.

2. Wählen Sie den zu verwendenden Snapshot aus

Führen Sie den Workflow aus "Listen Sie die Snapshots auf" Und wählen Sie den gewünschten Snapshot aus.

3. Die App klonen

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Account/{AccountID}/k8s/v1/manageApps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
JSON	Text	Ja.	Stellt die Parameter für die geklonte App bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

```
{
  "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 Beispiel: Klonen einer App aus einem Snapshot

```
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
```

Klonen einer gemanagten Applikation aus einem Backup

Sie können eine neue gemanagte Applikation erstellen, indem Sie diese über ein Applikations-Backup klonen.

Bevor Sie beginnen

Beachten Sie Folgendes zu diesem Workflow:

- Es wird ein App-Backup verwendet
- Der Klonvorgang wird im selben Cluster durchgeführt

> Zum Klonen einer App auf einem anderen Cluster müssen Sie den aktualisieren `clusterId` Parameter in den JSON-Input, wie es für Ihre Umgebung geeignet ist.

1. Wählen Sie die verwaltete App zum Klonen aus

Führen Sie den Workflow aus "Listen Sie die verwalteten Apps auf" Und wählen Sie die Anwendung aus, die Sie klonen möchten. Für DEN REST-Aufruf, der zum Klonen der App verwendet wird, sind mehrere Ressourcenwerte erforderlich.

2. Wählen Sie das zu verwendende Backup aus

Führen Sie den Workflow aus "Listen Sie die Backups auf" Und wählen Sie das gewünschte Backup aus.

3. Die App klonen

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
POST	/Account/{AccountID}/k8s/v1/manageApps

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
JSON	Text	Ja.	Stellt die Parameter für die geklonte App bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

```
{
  "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 Beispiel: Klonen einer Applikation aus einem Backup

```
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
```

Wiederherstellen einer gemanagten Applikation aus einem Backup

Sie können eine gemanagte Applikation wiederherstellen, indem Sie eine neue Applikation aus einem Backup erstellen.

1. Wählen Sie die verwaltete Anwendung, die wiederhergestellt werden soll

Führen Sie den Workflow aus "Listen Sie die verwalteten Apps auf" Und wählen Sie die Anwendung aus, die Sie klonen möchten. Für DEN REST-Aufruf, der zum Klonen der App verwendet wird, sind mehrere Ressourcenwerte erforderlich.

2. Wählen Sie das zu verwendende Backup aus

Führen Sie den Workflow aus "Listen Sie die Backups auf" Und wählen Sie das gewünschte Backup aus.

3. Stellen Sie die App wieder her

Führen Sie den folgenden REST-API-Aufruf aus. Sie müssen die ID für ein Backup (wie unten gezeigt) oder einen Snapshot angeben.

HTTP-Methode	Pfad
PUT	/Account/{AccountID}/k8s/v1/manageApps/{appID}

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
JSON	Text	Ja.	Stellt die Parameter für die geklonte App bereit. Siehe das folgende Beispiel.

JSON-Eingabebeispiel

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

Curl Beispiel: Wiederherstellen einer vorhandenen Applikation aus einem Backup

```
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
```

Unterstützung

Listen Sie die Benachrichtigungen auf

Sie können die Benachrichtigungen für ein bestimmtes Astra-Konto auflisten. Dies können Sie im Rahmen der Überwachung der Systemaktivität oder des Debugging eines Problems tun.

1. Listen Sie die Benachrichtigungen auf

Führen Sie den folgenden REST-API-Aufruf aus.

HTTP-Methode	Pfad
GET	/Account/{AccountID}/Core/v1/notifications

Zusätzliche Eingabeparameter

Zusätzlich zu den Parametern, die bei allen REST-API-Aufrufen üblich sind, werden die folgenden Parameter auch in den Curl-Beispielen für diesen Schritt verwendet.

Parameter	Typ	Erforderlich	Beschreibung
Filtern	Abfrage	Nein	Filtern Sie optional die Benachrichtigungen, die in der Antwort zurückgegeben werden sollen.
Einschließlich	Abfrage	Nein	Wählen Sie optional die Werte aus, die in der Antwort zurückgegeben werden sollen.

Curl Beispiel: Alle Benachrichtigungen zurückgeben

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

Curl Beispiel: Gibt die Beschreibung für Benachrichtigungen mit Schweregrad der Warnung zurück

```
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>'
```

Beispiel für eine JSON-Ausgabe

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


Löschen einer fehlgeschlagenen Anwendung

Sie können eine verwaltete Anwendung möglicherweise nicht entfernen, wenn eine Sicherung oder ein Snapshot in einem fehlgeschlagenen Zustand vorhanden ist. In diesem Fall können Sie die App mithilfe des unten beschriebenen Workflows manuell entfernen.

1. Wählen Sie die zu löschende verwaltete App aus

Führen Sie den Workflow aus "Listen Sie die verwalteten Apps auf" Und wählen Sie die Anwendung aus, die Sie entfernen möchten.

2. Listen Sie die bestehenden Backups für die App auf

Führen Sie den Workflow aus "Listen Sie die Backups auf".

3. Löschen Sie alle Backups

Löschen Sie alle App-Backups durch Ausführen des Workflows "Löschen Sie ein Backup" Für jedes Backup in der Liste.

4. Listen Sie die vorhandenen Snapshots für die App auf

Führen Sie den Workflow aus "Listen Sie die Snapshots auf".

5. Löschen Sie alle Snapshots

Führen Sie den Workflow aus "Löschen Sie einen Snapshot" Von jedem Snapshot in der Liste.

6. Entfernen Sie die Anwendung

Führen Sie den Workflow aus "Verwaltung einer Anwendung aufheben" Um die Anwendung zu entfernen.

Verwendung Von Python

NetApp Astra Control Python SDK

NetApp Astra Control Python SDK ist ein Open-Source-Paket, mit dem Sie eine Astra Control Implementierung automatisieren können. Das Paket ist auch eine wertvolle Ressource, um sich über die Astra Control REST API zu informieren, vielleicht im Rahmen der Erstellung Ihrer eigenen Automatisierungsplattform.

Zur Einfachheit wird das NetApp Astra Control Python SDK durchgehend auf dieser Seite als **SDK** bezeichnet.

Zwei verwandte Softwaretools

Das SDK enthält zwei verschiedene, wenn auch verwandte Tools, die auf verschiedenen Ebenen der Abstraktion beim Zugriff auf die Astra Control REST API arbeiten.

Astra SDK

Das Astra SDK bietet die Kernfunktionen der Plattform. Es enthält eine Reihe von Python-Klassen, in denen die zugrunde liegenden REST-API-Aufrufe abstrahiert werden. Die Klassen unterstützen administrative Aktionen auf verschiedenen Astra Control Ressourcen, einschließlich Apps, Backups, Snapshots und Cluster.

Das Astra SDK ist ein Teil des Pakets und wird in der Single bereitgestellt `astraSDK.py` Datei: Sie können diese Datei in Ihre Umgebung importieren und die Klassen direkt verwenden.

Das **NetApp Astra Control Python SDK** (oder nur SDK) ist der Name des gesamten Pakets. Das **Astra SDK** bezieht sich auf die Core Python Klassen in der einzelnen Datei `astraSDK.py`.

Toolkit-Skript

Neben der Astra SDK-Datei, die `toolkit.py` Skript ist ebenfalls verfügbar. Dieses Skript wird auf einer höheren Abstraktionsebene ausgeführt und bietet Zugriff auf diskrete, intern als Python-Funktionen definierte administrative Aktionen. Das Skript importiert das Astra SDK und ruft nach Bedarf die Klassen an.

Zugang zu

Sie haben folgende Möglichkeiten, auf das SDK zuzugreifen:

Python-Paket

Das SDK ist unter verfügbar "Python-Paketindex" Unter dem Namen **netapp-astra-Toolkits**. Dem Paket wird eine Versionsnummer zugewiesen und bei Bedarf auch weiterhin aktualisiert. Sie müssen das Paketverwaltungsprogramm * PIP* verwenden, um das Paket in Ihrer Umgebung zu installieren.

Siehe "PyPI: NetApp Astra Control Python SDK" Finden Sie weitere Informationen.

GitHub-Quellcode

Der SDK-Quellcode ist auch bei GitHub erhältlich. Das Repository umfasst Folgendes:

- `astraSDK.py` (Astra SDK mit Python-Klassen)
- `toolkit.py` (Auf höherer Ebene funktionbasiertes Skript)

- Detaillierte Installationsanforderungen und Anweisungen
- Installationsskripte
- Zusätzliche Dokumentation

Sie können die klonen "GitHub: NetApp/netapp-astra-Toolkits" Repository in Ihre lokale Umgebung einbinden

Installation und grundlegende Anforderungen

Es gibt verschiedene Optionen und Anforderungen, die bei der Installation des Pakets und bei der Vorbereitung der Verwendung berücksichtigt werden müssen.

Zusammenfassung der Installationsoptionen

Sie können das SDK auf eine der folgenden Arten installieren:

- Verwenden Sie Pip, um das Paket von PyPI in Ihre Python-Umgebung zu installieren
- Git Hub-Repository klonen und entweder:
  - Implementieren des Pakets als Docker Container und umfasst alle erforderlichen Komponenten.
  - Kopieren Sie die beiden Core Python-Dateien, damit sie für Ihren Python-Client-Code zugänglich sind

Weitere Informationen finden Sie auf den Seiten PyPI und GitHub.

Anforderungen für die Astra Control-Umgebung

Ob direkt die Python-Klassen im Astra SDK oder die Funktionen im `toolkit.py` Skript, schließlich sind Sie bei einer Implementierung von Astra Control auf DIE REST-API zugreifen. Aus diesem Grund benötigen Sie ein Astra-Konto zusammen mit einem API-Token. Siehe "Bevor Sie beginnen" Und die anderen Seiten im Abschnitt **Get Started** dieser Dokumentation für weitere Informationen.

Anforderungen für das NetApp Astra Control Python SDK

Das SDK verfügt über mehrere Voraussetzungen für die lokale Python-Umgebung. Beispiel: Sie müssen Python 3.5 oder höher verwenden. Darüber hinaus sind mehrere Python-Pakete erforderlich. Weitere Informationen finden Sie auf der GitHub Repository-Seite oder auf der Seite des PyPI-Pakets.

Zusammenfassung hilfreicher Ressourcen

Im Folgenden finden Sie einige Ressourcen, die Sie für den Einstieg benötigen.

- "PyPI: NetApp Astra Control Python SDK"
- "GitHub: NetApp/netapp-astra-Toolkits"

Native Python

Bevor Sie beginnen

Python ist eine beliebte Entwicklungssprache speziell für die Rechenzentrumsautomatisierung. Bevor Sie die nativen Funktionen von Python zusammen mit mehreren gängigen Paketen nutzen, müssen Sie die Umgebung und die erforderlichen Eingabedateien vorbereiten.

NetApp hat nicht nur direkt mit Python auf die Astra Control REST API zugegriffen, sondern bietet auch ein Toolkit-Paket, das die API abstrahiert und einige der Komplexität beseitigt. Siehe "NetApp Astra Control Python SDK" Finden Sie weitere Informationen.

Bereiten Sie die Umgebung vor

Im Folgenden werden die grundlegenden Konfigurationsanforderungen für die Ausführung der Python-Skripte beschrieben.

Python 3

Sie müssen die neueste Version von Python 3 installiert haben.

Weitere Bibliotheken

Die Bibliotheken **Requests** und **urllib3** müssen installiert sein. Sie können je nach Ihrer Umgebung Pip oder ein anderes Python Management Tool verwenden.

Netzwerkzugriff

Die Arbeitsstation, auf der die Skripte ausgeführt werden, muss Netzwerkzugriff haben und Astra Control erreichen können. Bei der Verwendung des Astra Control Service müssen Sie mit dem Internet verbunden sein und eine Verbindung zum Dienst unter herstellen können `https://astra.netapp.io`.

Identitätsinformationen

Sie benötigen ein gültiges Astra-Konto mit der Account-ID und dem API-Token. Siehe "Holen Sie sich ein API-Token" Finden Sie weitere Informationen.

Erstellen Sie die JSON-Eingabedateien

Die Python-Skripte basieren auf Konfigurationsinformationen in JSON-Eingabedateien. Im Folgenden finden Sie Beispieldateien.

Sie müssen die Proben entsprechend Ihrer Umgebung aktualisieren.

Identitätsinformationen

Die folgende Datei enthält das API-Token und das Astra-Konto. Sie müssen diese Datei mit der an Python-Skripte übergeben `-i` (Oder `--identity`) CLI-Parameter.

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

Listen Sie die verwalteten Apps auf

Mit dem folgenden Skript können Sie die verwalteten Anwendungen für Ihr Astra-Konto auflisten.

Siehe "Bevor Sie beginnen" Beispiel für die erforderliche JSON-Eingabedatei.

```
#!/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-Referenz

Sie können auf die Details aller Astra Control REST-API-Aufrufe zugreifen, einschließlich HTTP-Methoden, Eingabeparameter und Antworten. Diese vollständige Referenz ist hilfreich, wenn Automatisierungsapplikationen mithilfe der REST API entwickelt werden.

Die REST-API-Referenzdokumentation wird derzeit mit Astra Control bereitgestellt und ist online verfügbar.

Bevor Sie beginnen

Sie benötigen ein Konto für Astra Control Center oder Astra Control Service.

Schritte

1. Melden Sie sich mit Ihren Anmeldedaten im Astra an.

   Rufen Sie die folgende Website für den Astra Control Service auf: "https://astra.netapp.io"

2. Klicken Sie auf das Figurensymbol oben rechts auf der Seite und wählen Sie **API Access**.
3. Klicken Sie oben auf der Seite auf die URL, die unter **API Documentation** angezeigt wird.
4. Geben Sie bei Aufforderung erneut Ihre Anmeldeinformationen für das Konto an.

Weitere Ressourcen

Auf weitere Ressourcen erhalten Sie Zugriff. Dort erhalten Sie weitere Informationen zu NetApp Cloud Services und Support sowie ZU allgemeinen REST- und Cloud-Konzepten.

Astra

- "Astra Control Center 22.04-Dokumentation"

  Dokumentation für die aktuelle Version der Astra Control Center-Software, die beim Kunden vor Ort eingesetzt wird.

- "Dokumentation des Astra Control Service"

  Dokumentation für die aktuelle Version der Astra Control Service-Software, die in der Public Cloud verfügbar ist.

- "Astra Trident-Dokumentation"

  Dokumentation für die aktuelle Version der Astra Trident Software, einem Open-Source-Storage-Orchestrator von NetApp.

- "Dokumentation der Astra-Familie"

  Zentraler Standort für den Zugriff auf die gesamte Astra Dokumentation für On-Premises- und Public-Cloud-Implementierungen.

NetApp Cloud-Ressourcen

- "NetApp Cloud-Lösungen"

  Zentraler Standort für NetApp Cloud Lösungen.

- "NetApp Cloud Central Konsole"

  NetApp Cloud Central Service-Konsole mit Anmeldung.

- "NetApp Support"

  Sie erhalten Zugriff auf Tools für die Fehlerbehebung, Dokumentation und technische Support-Unterstützung.

REST- und Cloud-Konzepte

- PhD "Dissertation" Von Roy Fielding

  In dieser Publikation wurde das MODELL DER REST-Anwendungsentwicklung eingeführt und etabliert.

- "Auth0"

  Dies ist der Authentifizierungs- und Autorisierungsplattform-Service, der vom Astra-Service für den

Webzugriff genutzt wird.

- "RFC-Editor"

  Maßgebliche Quelle für Web- und Internet-Standards wird als Sammlung von eindeutig nummerierten RFC-Dokumenten gepflegt.

Frühere Versionen der Dokumentation Astra Control Automation

Die Dokumentation zur Automatisierung früherer Astra Control Versionen finden Sie unter den nachfolgenden Links.

- "Astra Control Automation 21.12 - Dokumentation"
- "Astra Control Automation 21.08 - Dokumentation"

Rechtliche Hinweise

Rechtliche Hinweise ermöglichen den Zugriff auf Copyright-Erklärungen, Marken, Patente und mehr.

Urheberrecht

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

Marken

NetApp, das NETAPP Logo und die auf der NetApp Markenseite aufgeführten Marken sind Marken von NetApp Inc. Andere Firmen- und Produktnamen können Marken der jeweiligen Eigentümer sein.

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

Patente

Eine aktuelle Liste der NetApp Patente finden Sie unter:

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

Datenschutzrichtlinie

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

Astra Control API-Lizenz

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