基础架构工作流

Astra Automation

NetApp
August 11, 2025

目录

基础架构工作流

准备使用基础架构工作流

您可以使用这些工作流创建和维护用于Astra Control Center部署的基础架构。在许多情况下、这些工作流也可以与Astra Control Service结合使用。

NetApp 可以随时扩展和改进这些工作流，因此您应定期查看这些工作流。

一般准备

在使用任何 Astra 工作流之前，请务必查看 "准备使用这些工作流"。

工作流类别

基础架构工作流按不同类别进行组织，以便更容易找到所需的工作流。

类别	Description
身份和访问	通过这些工作流，您可以管理身份以及如何访问 Astra 。这些资源包括用户，凭据和令牌。
LDAP 配置	您可以选择将Astra控制中心配置为使用LDAP对选定用户进行身份验证。
集群	您可以添加受管 Kubernetes 集群，以便保护和支持这些集群所包含的应用程序。
云	通过这些工作流、可以访问通过Astra Control REST API提供的云。
存储分段	您可以使用这些工作流创建和管理用于存储备份的 S3 存储分段。
存储	通过这些工作流，您可以添加和维护存储后端和卷。

身份和访问

列出用户

您可以列出为特定 Astra 帐户定义的用户。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /core/v1/users

其他输入参数

除了所有 REST API 调用通用的参数之外，此步骤的 curl 示例还使用以下参数。

参数	Type	Required	Description
包括	查询	否	也可以选择要在响应中返回的值。

curl 示例：返回所有用户的所有数据

```
curl --request GET \
--location "https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/users" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

CURL示例：返回所有用户的名字、姓氏和**ID**

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/users?include=firstN
ame,lastName,id" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 输出示例

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

创建用户

您可以使用特定凭据和预定义角色创建用户。您也可以选择限制用户对特定命名空间的访问。

第1步：选择用户名

执行工作流 "列出用户" 并选择当前未使用的可用名称。

第2步：创建用户

执行以下REST API调用以创建用户。成功完成调用后、新用户将无法使用。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/users

curl 示例

```
curl --request POST \
--location "https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/users" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-user",
  "version" : "1.1",
  "firstName" : "John",
  "lastName" : "West",
  "email" : "jwest@example.com"
}
```

JSON 输出示例


```
{
    "metadata": {
        "creationTimestamp": "2022-11-20T17:23:15Z",
        "modificationTimestamp": "2022-11-20T17:23:15Z",
        "createdBy": "a20e91f3-2c49-443b-b240-615d940ec5f3",
        "labels": []
    },
    "type": "application/astra-user",
    "version": "1.2",
    "id": "d07dac0a-a328-4840-a216-12de16bbd484",
    "authProvider": "local",
    "authID": "jwest@example.com",
    "firstName": "John",
    "lastName": "West",
    "companyName": "",
    "email": "jwest@example.com",
    "postalAddress": {
        "addressCountry": "",
        "addressLocality": "",
        "addressRegion": "",
        "streetAddress1": "",
        "streetAddress2": "",
        "postalCode": ""
    },
    "state": "active",
    "sendWelcomeEmail": "false",
    "isEnabled": "true",
    "isInviteAccepted": "true",
    "enableTimestamp": "2022-11-20T17:23:15Z",
    "lastActTimestamp": ""
}
```


第**3**步：**(**可选**)**选择允许的名称空格

执行工作流 "列出命名空间" 并选择要限制访问的命名空间。

第**4**步：将用户绑定到角色

执行以下REST API调用以将用户绑定到角色。以下示例对命名空间访问没有任何限制。请参见 "具有命名空间粒度的增强型 RBAC" 有关详细信息 …

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/roleBindings

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/roleBindings" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-roleBinding",
  "version" : "1.1",
  "userID" : "d07dac0a-a328-4840-a216-12de16bbd484",
  "accountID" : "29e1f39f-2bf4-44ba-a191-5b84ef414c95",
  "role" : "viewer",
  "roleConstraints": [ "*" ]
}
```

第**5**步：创建凭据

执行以下REST API调用以创建凭据并将其与用户关联。此示例使用作为base64值提供的密码。。 `name` 属性应包含上一步返回的用户的ID。输入属性 `change` 还必须在base64中进行编码、并确定用户是否必须在首次登录时更改其密码 (`true` 或 `false`）。

只有使用本地身份验证部署Astra控制中心时、才需要执行此步骤。使用LDAP部署Astra控制中心或部署Astra控制服务时不需要此功能。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/credentials

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/credentials" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-credential",
  "version" : "1.1",
  "name" : "d07dac0a-a328-4840-a216-12de16bbd484",
  "keyType" : "passwordHash",
  "keyStore" : {
      "cleartext" : "TmV0QXBwMTIz",
      "change" : "ZmFsc2U="
  },
  "valid" : "true"
}
```

LDAP 配置

准备LDAP配置

您可以选择将Astra控制中心与轻型目录访问协议(Lightweight Directory Access Protocol 、LDAP)服务器集成、以便为选定的Astra用户执行身份验证。LDAP是一种用于访问分布式目录信息的行业标准协议、也是企业身份验证的常见选择。

相关信息

- "LDAP技术规格路线图"
- "LDAP版本3"

实施过程概述

总体而言、要配置LDAP服务器以为Astra用户提供身份验证、需要执行几个步骤。

> 尽管下面介绍的步骤是按顺序执行的、但在某些情况下、您可以按不同顺序执行这些步骤。例如、您可以在配置LDAP服务器之前定义Astra用户和组。

1. 请查看 "要求和限制" 了解选项、要求和限制。

2. 选择LDAP服务器和所需的配置选项(包括安全性)。
3. 执行工作流 "将Astra配置为使用LDAP服务器" 将Astra与LDAP服务器集成。
4. 查看LDAP服务器上的用户和组、确保它们定义正确。
5. 在中执行相应的工作流 "将LDAP条目添加到Astra" 确定要使用LDAP进行身份验证的用户。

要求和限制

在将Astra配置为使用LDAP进行身份验证之前、您应查看下面介绍的Astra配置要点、包括限制和配置选项。

仅支持**Astra**控制中心

Astra Control平台提供了两种部署模式。只有Astra控制中心部署才支持LDAP身份验证。

使用**REST API**或**Web**用户界面进行配置

当前版本的Astra控制中心支持使用Astra Control REST API以及Astra Web用户界面配置LDAP身份验证。

需要**LDAP**服务器

要接受和处理Astra身份验证请求、您必须具有LDAP服务器。当前版本的Astra控制中心支持Microsoft的Active Directory。

与**LDAP**服务器的安全连接

在Astra中配置LDAP服务器时、您可以选择定义安全连接。在这种情况下、LDAPS协议需要证书。

配置用户或组

您需要选择要使用LDAP进行身份验证的用户。为此、您可以确定各个用户或一组用户。必须在LDAP服务器上定义这些帐户。它们还需要在Astra (类型为LDAP)中进行标识、这样可以将身份验证请求转发到LDAP。

绑定用户或组时的角色限制

在当前版本的Astra控制中心中、`roleConstrcont`唯一支持的值是"*"。这表示用户不受限于一组有限的命名空间、并且可以访问所有命名空间。请参见 "将LDAP条目添加到Astra" 有关详细信息 …

LDAP凭据

LDAP使用的凭据包括用户名(电子邮件地址)和关联的密码。

唯一电子邮件地址

在Astra控制中心部署中用作用户名的所有电子邮件地址都必须是唯一的。您不能添加电子邮件地址已定义为Astra的LDAP用户。如果存在重复的电子邮件、您需要先从Astra中将其删除。请参见 "删除用户" 有关详细信息、请访问Astra控制中心文档站点。

也可以先定义**LDAP**用户和组

您可以将LDAP用户和组添加到Astra控制中心、即使它们尚未位于LDAP中或未配置LDAP服务器也是如此。这样、您可以在配置LDAP服务器之前预先配置用户和组。

在多个**LDAP**组中定义的用户

如果某个LDAP用户属于多个LDAP组、并且在Astra中为这些组分配了不同的角色、则用户在进行身份验证时的有效角色将获得最大的特权。例如、如果为用户分配了组1的`viewer`角色、但在组2中具有`mMember`角色、则该用户的角色将为`mMember`。这基于Astra使用的层次结构(从高到低):

- 所有者

- 管理员
- 成员
- 查看器

定期同步帐户

Astra大约每60秒就会将其用户和组与LDAP服务器同步一次。因此、如果将用户或组添加到LDAP或从LDAP中删除、则可能需要长达一分钟的时间、才能在Astra中使用该用户或组。

禁用并重置**LDAP**配置

在尝试重置LDAP配置之前、必须先禁用LDAP身份验证。此外、要更改LDAP服务器(`connectionHost`)、您需要同时执行这两个操作。请参见 "禁用并重置LDAP" 有关详细信息 …

REST API参数

LDAP配置工作流会调用REST API来完成特定任务。每个API调用都可以包括输入参数、如提供的示例所示。请参见 "联机API参考" 有关如何查找参考文档的信息。

将**Astra**配置为使用**LDAP**服务器

您需要选择LDAP服务器并将Astra配置为使用该服务器作为身份验证提供程序。配置任务包括以下步骤。每个步骤都包括一个REST API调用。

第**1**步：添加**CA**证书

执行以下REST API调用、将CA证书添加到Astra。

此步骤是可选的、只有当您希望Astra和LDAP通过使用LDAPS的安全通道进行通信时、才需要执行此步骤。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/certificates

curl 示例

```
curl --request POST \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/certificates" \
--include \
--header "Content-Type: application/astra-certificate+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-certificate",
  "version": "1.0",
  "certUse": "rootCA",
  "cert": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMyVEN",
  "isSelfSigned": "true"
}
```

有关输入参数、请注意以下事项：

- `cert`是一个JSON字符串、其中包含base64编码的PKCS-11格式证书(PEM编码)。
- 如果证书为自签名证书、则应将`isSelfSigned`设置为`true`。默认值为 `false` 。

JSON 输出示例


```
{
  "type": "application/astra-certificate",
  "version": "1.0",
  "id": "a5212e7e-402b-4cff-bba0-63f3c6505199",
  "certUse": "rootCA",
  "cert": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMyVEN",
  "cn": "adldap.example.com",
  "expiryTimestamp": "2023-07-08T20:22:07Z",
  "isSelfSigned": "true",
  "trustState": "trusted",
  "trustStateTransitions": [
  {
    "from": "untrusted",
    "to": [
      "trusted",
      "expired"
    ]
  },
  {
    "from": "trusted",
    "to": [
      "untrusted",
      "expired"
    ]
  },
  {
    "from": "expired",
    "to": [
     "untrusted",
     "trusted"
    ]
  }
  ],
  "trustStateDesired": "trusted",
  "trustStateDetails": [],
  "metadata": {
    "creationTimestamp": "2022-07-21T04:16:06Z",
    "modificationTimestamp": "2022-07-21T04:16:06Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "modifiedBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
    }
}
```

第2步：添加绑定凭据

执行以下REST API调用以添加绑定凭据。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/credentials

curl 示例

```
curl --request POST \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/credentials" \
--include \
--header "Content-Type: application/astra-certificate+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "name": "ldapBindCredential",
  "type": "application/astra-credential",
  "version": "1.1",
  "keyStore": {
    "bindDn": "dWlkPWFkbWluLG91PXN5c3RlbQ==",
    "password": "cGFzc3dvcmQ="
  }
}
```

有关输入参数、请注意以下事项：

- `binddn`和`password`是LDAP管理员用户的base64编码绑定凭据、可连接和搜索LDAP目录。`binddn`是LDAP用户的电子邮件地址。

JSON 输出示例

```
{
    "type": "application/astra-credential",
    "version": "1.1",
    "id": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "name": "ldapBindCredential",
    "metadata": {
        "creationTimestamp": "2022-07-21T06:53:11Z",
        "modificationTimestamp": "2022-07-21T06:53:11Z",
        "createdBy": "527329f2-662c-41c0-ada9-2f428f14c137"
    }
}
```

请注意以下响应参数：

- 在后续工作流步骤中使用凭据的`id`。

第3步：检索LDAP设置的UUID

执行以下REST API调用以检索Astra控制中心附带的`Astra.account.ldap`设置的UUID。

以下cURL示例使用查询参数筛选设置收集。您可以改为删除此筛选器以获取所有设置、然后搜索`Astra.account.ldap`。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /core/v1/settings

curl 示例

```
curl --request GET \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/settings?filter=na
me%20eq%20'astra.account.ldap'&include=name,id" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
```

JSON 输出示例

```
{
  "items": [
    ["astra.account.ldap",
    "12072b56-e939-45ec-974d-2dd83b7815df"
    ]
  ],
  "metadata": {}
}
```

第**4**步：更新**LDAP**设置

执行以下REST API调用以更新LDAP设置并完成配置。使用上一个API调用中的`id`值作为以下URL路径中的`<setting_ID>`值。

> 您可以先对特定设置的GET请求进行问题描述 处理、以查看configSchema。此操作将提供有关配置中所需字段的详细信息。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
PUT	/accouns/ {account_id} /core/v1/settings/ {setting_id}

curl 示例

```
curl --request PUT \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/settings/<SETTING_
ID>" \
--include \
--header "Content-Type: application/astra-setting+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "myldap.example.com",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "true",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
    }
}
```

有关输入参数、请注意以下事项：

- `isenabled`应设置为`true`、否则可能会发生错误。
- `credentialId`是先前创建的绑定凭据的ID。
- `secureMode` should be set to `LDAP` or `LDAPS` based on your configuration in the earlier stee.
- 仅支持使用"Active Directory"作为供应商。

如果调用成功、则返回HTTP 204响应。

第5步：检索LDAP设置

您可以选择执行以下REST API调用来检索LDAP设置并确认更新。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /core/v1/settings/ {setting_id}

curl 示例

```
curl --request GET \
--location
"'https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/settings/<SETTING
_ID>" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 输出示例


```
{
  "items": [
  {
    "type": "application/astra-setting",
    "version": "1.0",
    "metadata": {
      "creationTimestamp": "2022-06-17T21:16:31Z",
      "modificationTimestamp": "2022-07-21T07:12:20Z",
      "labels": [],
      "createdBy": "system",
      "modifiedBy": "00000000-0000-0000-0000-000000000000"
    },
    "id": "12072b56-e939-45ec-974d-2dd83b7815df",
    "name": "astra.account.ldap",
    "desiredConfig": {
      "connectionHost": "10.193.61.88",
      "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
      "groupBaseDN": "ou=groups,ou=astra,dc=example,dc=com",
      "isEnabled": "true",
      "port": 686,
      "secureMode": "LDAPS",
      "userBaseDN": "ou=users,ou=astra,dc=example,dc=com",
      "userSearchFilter": "((objectClass=User))",
      "vendor": "Active Directory"
    },
    "currentConfig": {
      "connectionHost": "10.193.160.209",
      "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
      "groupBaseDN": "ou=groups,ou=astra,dc=example,dc=com",
      "isEnabled": "true",
      "port": 686,
      "secureMode": "LDAPS",
      "userBaseDN": "ou=users,ou=astra,dc=example,dc=com",
      "userSearchFilter": "((objectClass=User))",
```

```
      "vendor": "Active Directory"
    },
    "configSchema": {
      "$schema": "http://json-schema.org/draft-07/schema#",
      "title": "astra.account.ldap",
      "type": "object",
      "properties": {
        "connectionHost": {
          "type": "string",
          "description": "The hostname or IP address of your LDAP server."
        },
        "credentialId": {
          "type": "string",
          "description": "The credential ID for LDAP account."
        },
        "groupBaseDN": {
          "type": "string",
          "description": "The base DN of the tree used to start the group
search. The system searches the subtree from the specified location."
        },
        "groupSearchCustomFilter": {
          "type": "string",
          "description": "Type of search that controls the default group
search filter used."
        },
        "isEnabled": {
          "type": "string",
          "description": "This property determines if this setting is
enabled or not."
        },
        "port": {
          "type": "integer",
          "description": "The port on which the LDAP server is running."
        },
        "secureMode": {
          "type": "string",
          "description": "The secure mode LDAPS or LDAP."
        },
        "userBaseDN": {
          "type": "string",
          "description": "The base DN of the tree used to start the user
search. The system searches the subtree from the specified location."
        },
        "userSearchFilter": {
          "type": "string",
          "description": "The filter used to search for users according a
```

```
search criteria."
        },
        "vendor": {
          "type": "string",
          "description": "The LDAP provider you are using.",
          "enum": ["Active Directory"]
        }
      },
      "additionalProperties": false,
      "required": [
        "connectionHost",
        "secureMode",
        "credentialId",
        "userBaseDN",
        "userSearchFilter",
        "groupBaseDN",
        "vendor",
        "isEnabled"
      ]
      },
      "state": "valid",
    }
  ],
  "metadata": {}
}
```

在响应中找到`state`字段、该字段将具有下表中的一个值。

State	Description
待定	配置过程仍处于活动状态、尚未完成。
valid	已成功完成配置、并且响应中的`currentConfig`匹配`desiredConfig`。
error	LDAP配置过程失败。

将LDAP条目添加到Astra

将LDAP配置为Astra控制中心的身份验证提供程序后、您可以选择Astra将使用LDAP凭据进行身份验证的LDAP用户。每个用户都必须在Astra中具有一个角色、然后才能通过Astra Control REST API访问Astra。

您可以通过两种方式配置Astra来分配角色。选择适合您的环境的选项。

- "添加并绑定单个用户"
- "添加和绑定组"

LDAP凭据以用户名作为电子邮件地址以及关联的LDAP密码的形式提供。

添加并绑定单个用户

您可以为每个Astra用户分配一个角色、该角色将在LDAP身份验证后使用。如果用户数量较少、并且每个用户的管理特征可能不同、则这种做法是合适的。

第1步：添加用户

执行以下REST API调用、将用户添加到Astra并指示LDAP是身份验证提供程序。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/users

curl 示例

```
curl --request POST \
--location "https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/users"
\
--include \
--header "Content-Type: application/astra-user+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-user",
  "version" : "1.1",
  "authID" : "cn=JohnDoe,ou=users,ou=astra,dc=example,dc=com",
  "authProvider" : "ldap",
  "firstName" : "John",
  "lastName" : "Doe",
  "email" : "john.doe@example.com"
}
```

有关输入参数、请注意以下事项：

- 需要以下参数：
  - authProvider
  - authId

- 电子邮件

- `authId`是LDAP中用户的可分辨名称(DN)
- `email`对于在Astra中定义的所有用户都必须是唯一的

如果`email`值不唯一、则会发生错误、并在响应中返回409 HTTP状态代码。

JSON 输出示例


```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T17:44:18Z",
    "modificationTimestamp": "2022-07-21T17:44:18Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  },
  "type": "application/astra-user",
  "version": "1.2",
  "id": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "authProvider": "ldap",
  "authID": "cn=JohnDoe,ou=users,ou=astra,dc=example,dc=com",
  "firstName": "John",
  "lastName": "Doe",
  "companyName": "",
  "email": "john.doe@example.com",
  "postalAddress": {
    "addressCountry": "",
    "addressLocality": "",
    "addressRegion": "",
    "streetAddress1": "",
    "streetAddress2": "",
    "postalCode": ""
  },
  "state": "active",
  "sendWelcomeEmail": "false",
  "isEnabled": "true",
  "isInviteAccepted": "true",
  "enableTimestamp": "2022-07-21T17:44:18Z",
  "lastActTimestamp": ""
}
```


第2步：为用户添加角色绑定

执行以下REST API调用以将用户绑定到特定角色。您需要具有上一步中创建的用户UUID。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/roleBindings

curl 示例

```
curl --request POST \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/roleBindings" \
--include \
--header "Content-Type: application/astra-roleBinding+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-roleBinding",
  "version": "1.1",
  "accountID": "{account_id}",
  "userID": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "role": "member",
  "roleConstraints": ["*"]
}
```

有关输入参数、请注意以下事项：

- 以上用于`roleConstrcont`的值是当前版本Astra唯一可用的选项。它表示用户不受限于一组有限的命名空间、并且可以访问所有这些命名空间。

JSON响应示例

```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T18:08:24Z",
    "modificationTimestamp": "2022-07-21T18:08:24Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  },
  "type": "application/astra-roleBinding",
  "principalType": "user",
  "version": "1.1",
  "id": "b02c7e4d-d483-40d1-aaff-e1f900312114",
  "userID": "a7b5e674-a1b1-48f6-9729-6a571426d49f",
  "groupID": "00000000-0000-0000-0000-000000000000",
  "accountID": "d0fdbfa7-be32-4a71-b59d-13d95b42329a",
  "role": "member",
  "roleConstraints": ["*"]
}
```

请注意以下有关响应参数的信息：

- `PrincipalType`字段的值`user`表示已为用户(而不是组)添加角色绑定。

添加和绑定组

您可以为Astra组分配一个角色、该角色将在LDAP身份验证后使用。如果用户数量很多、并且每个用户都可能具有类似的管理特征、则这种做法是合适的。

第1步：添加组

执行以下REST API调用、将组添加到Astra并指示LDAP是身份验证提供程序。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/｛account_id｝/core/v1/groups

curl 示例

```
curl --request POST \
--location "https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/groups"
\
--include \
--header "Content-Type: application/astra-group+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-group",
  "version": "1.0",
  "name": "Engineering",
  "authProvider": "ldap",
  "authID": "CN=Engineering,OU=groups,OU=astra,DC=example,DC=com"
}
```

有关输入参数、请注意以下事项：

- 需要以下参数：
  - authProvider
  - authId

JSON响应示例

```
{
  "type": "application/astra-group",
  "version": "1.0",
  "id": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "name": "Engineering",
  "authProvider": "ldap",
  "authID": "CN=Engineering,OU=groups,OU=astra,DC=example,DC=com",
  "metadata": {
    "creationTimestamp": "2022-07-21T18:42:52Z",
    "modificationTimestamp": "2022-07-21T18:42:52Z",
    "createdBy": "8a02d2b8-a69d-4064-827f-36851b3e1e6e",
    "labels": []
  }
}
```

第2步：为组添加角色绑定

执行以下REST API调用以将组绑定到特定角色。您需要具有上一步中创建的组的UUID。在LDAP执行身份验证后、属于组成员的用户将能够登录到Astra。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/roleBindings

curl 示例

```
curl --request POST \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/roleBindings" \
--include \
--header "Content-Type: application/astra-roleBinding+json" \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-roleBinding",
  "version": "1.1",
  "accountID": "{account_id}",
  "groupID": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "role": "viewer",
  "roleConstraints": ["*"]
}
```

有关输入参数、请注意以下事项：

- 以上用于`roleConstrcont`的值是当前版本Astra唯一可用的选项。它表示用户不受特定命名空间的限制、并且可以访问所有命名空间。

JSON响应示例

```
{
  "metadata": {
    "creationTimestamp": "2022-07-21T18:59:43Z",
    "modificationTimestamp": "2022-07-21T18:59:43Z",
    "createdBy": "527329f2-662c-41c0-ada9-2f428f14c137",
    "labels": []
  },
  "type": "application/astra-roleBinding",
  "principalType": "group",
  "version": "1.1",
  "id": "2f91b06d-315e-41d8-ae18-7df7c08fbb77",
  "userID": "00000000-0000-0000-0000-000000000000",
  "groupID": "8b5b54da-ae53-497a-963d-1fc89990525b",
  "accountID": "d0fdbfa7-be32-4a71-b59d-13d95b42329a",
  "role": "viewer",
  "roleConstraints": ["*"]
}
```

请注意以下有关响应参数的信息：

- `PrincipalType`字段的值`group`表示已为组(而不是用户)添加角色绑定。

禁用并重置LDAP

您可以根据需要为Astra控制中心部署执行两项可选的相关管理任务。您可以全局禁用LDAP身份验证并重置LDAP配置。

这两个工作流任务都需要`Astra.account.ldap` Astra设置的ID。有关如何检索设置ID的详细信息、请参见*配置LDAP服务器*。请参见 "检索LDAP设置的UUID" 有关详细信息 …

- "禁用LDAP身份验证"
- "重置LDAP身份验证配置"

禁用**LDAP**身份验证

您可以执行以下REST API调用来全局禁用特定Astra部署的LDAP身份验证。此调用会更新`Astra.account.ldap`设置、并且`isenabled`值设置为`false`。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
PUT	/accouns/ {account_id} /core/v1/settings/ {setting_id}

```
curl --request PUT \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/settings/<SETTING_
ID>" \
--include \
--header "Content-Type: application/astra-setting+json"
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "myldap.example.com",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "false",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
    }
}
```

如果调用成功、则返回`HTTP 204`响应。您可以选择再次检索配置设置以确认更改。

重置LDAP身份验证配置

您可以执行以下REST API调用以断开Astra与LDAP服务器的连接、并在Astra中重置LDAP配置。此调用将更新`Astra.account.ldap`设置、并清除`connectionHost`的值。

`isenabled`的值也必须设置为`false`。您可以在进行重置调用之前设置此值、也可以在进行重置调用时设置此值。在第二种情况下、应在同一重置调用中清除`connectionHost`并将`isenabled`设置为false。

> 此操作会造成系统中断、您应谨慎操作。它会删除所有已导入的LDAP用户和组。它还会删除您在Astra控制中心中创建的所有相关Astra用户、组和roleBindings (LDAP类型)。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
PUT	/accouns/ {account_id} /core/v1/settings/ {setting_id}

```
curl --request PUT \
--location
"https://astra.example.com/accounts/$ACCOUNT_ID/core/v1/settings/<SETTING_
ID>" \
--include \
--header "Content-Type: application/astra-setting+json"
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-setting",
  "version": "1.0",
  "desiredConfig": {
    "connectionHost": "",
    "credentialId": "3bd9c8a7-f5a4-4c44-b778-90a85fc7d154",
    "groupBaseDN": "OU=groups,OU=astra,DC=example,DC=com",
    "isEnabled": "false",
    "port": 686,
    "secureMode": "LDAPS",
    "userBaseDN": "OU=users,OU=astra,DC=example,dc=com",
    "userSearchFilter": "((objectClass=User))",
    "vendor": "Active Directory"
    }
}
```

请注意以下事项：

- 要更改LDAP服务器、必须禁用LDAP Changing `connectHost`并将其重置为空值、如上例所示。
- 如果调用成功、则返回`HTTP 204`响应。您也可以选择重新检索配置以确认更改。

集群

列出集群

您可以列出特定云中的可用集群。

第1步：选择云

执行工作流 "列出云" 并选择包含集群的云。

第2步：列出集群

执行以下REST API调用以列出特定云中的集群。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /topology/v1/cloud / {clune_id} /集群

curl 示例：返回所有集群的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/clouds/<CLOUD_ID
>/clusters" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 输出示例

```
{
    "items": [
        {
            "type": "application/astra-cluster",
            "version": "1.1",
            "id": "7ce83fba-6aa1-4e0c-a194-26e714f5eb46",
            "name": "openshift-clstr-ol-07",
            "state": "running",
            "stateUnready": [],
            "managedState": "managed",
            "protectionState": "full",
            "protectionStateDetails": [],
            "restoreTargetSupported": "true",
            "snapshotSupported": "true",
            "managedStateUnready": [],
            "managedTimestamp": "2022-11-03T15:50:59Z",
            "inUse": "true",
            "clusterType": "openshift",
            "accHost": "true",
            "clusterVersion": "1.23",
```

```
            "clusterVersionString": "v1.23.12+6b34f32",
            "namespaces": [
                "default",
                "kube-node-lease",
                "kube-public",
                "kube-system",
                "metallb-system",
                "mysql",
                "mysql-clone1",
                "mysql-clone2",
                "mysql-clone3",
                "mysql-clone4",
                "netapp-acc-operator",
                "netapp-monitoring",
                "openshift",
                "openshift-apiserver",
                "openshift-apiserver-operator",
                "openshift-authentication",
                "openshift-authentication-operator",
                "openshift-cloud-controller-manager",
                "openshift-cloud-controller-manager-operator",
                "openshift-cloud-credential-operator",
                "openshift-cloud-network-config-controller",
                "openshift-cluster-csi-drivers",
                "openshift-cluster-machine-approver",
                "openshift-cluster-node-tuning-operator",
                "openshift-cluster-samples-operator",
                "openshift-cluster-storage-operator",
                "openshift-cluster-version",
                "openshift-config",
                "openshift-config-managed",
                "openshift-config-operator",
                "openshift-console",
                "openshift-console-operator",
                "openshift-console-user-settings",
                "openshift-controller-manager",
                "openshift-controller-manager-operator",
                "openshift-dns",
                "openshift-dns-operator",
                "openshift-etcd",
                "openshift-etcd-operator",
                "openshift-host-network",
                "openshift-image-registry",
                "openshift-infra",
                "openshift-ingress",
                "openshift-ingress-canary",
```

```
            "openshift-ingress-operator",
            "openshift-insights",
            "openshift-kni-infra",
            "openshift-kube-apiserver",
            "openshift-kube-apiserver-operator",
            "openshift-kube-controller-manager",
            "openshift-kube-controller-manager-operator",
            "openshift-kube-scheduler",
            "openshift-kube-scheduler-operator",
            "openshift-kube-storage-version-migrator",
            "openshift-kube-storage-version-migrator-operator",
            "openshift-machine-api",
            "openshift-machine-config-operator",
            "openshift-marketplace",
            "openshift-monitoring",
            "openshift-multus",
            "openshift-network-diagnostics",
            "openshift-network-operator",
            "openshift-node",
            "openshift-oauth-apiserver",
            "openshift-openstack-infra",
            "openshift-operator-lifecycle-manager",
            "openshift-operators",
            "openshift-ovirt-infra",
            "openshift-sdn",
            "openshift-service-ca",
            "openshift-service-ca-operator",
            "openshift-user-workload-monitoring",
            "openshift-vsphere-infra",
            "pcloud",
            "postgreql",
            "trident"
        ],
        "defaultStorageClass": "4bacbb3c-0727-4f58-b13c-3a2a069baf89",
        "cloudID": "4f1e1086-f415-4451-a051-c7299cd672ff",
        "credentialID": "7ffd7354-b6c2-4efa-8e7b-cf64d5598463",
        "isMultizonal": "false",
        "tridentManagedStateAllowed": [
            "unmanaged"
        ],
        "tridentVersion": "22.10.0",
        "apiServiceID": "98df44dc-2baf-40d5-8826-e198b1b40909",
        "metadata": {
            "labels": [
                {
                    "name": "astra.netapp.io/labels/read-
```

```
only/cloudName",
                        "value": "private"
                    }
                ],
                "creationTimestamp": "2022-11-03T15:50:59Z",
                "modificationTimestamp": "2022-11-04T14:42:32Z",
                "createdBy": "00000000-0000-0000-0000-000000000000"
            }
        }
    ]
}
```

使用凭据添加集群

您可以添加一个集群、使其可由Astra管理。从Astra 22.11版开始、您可以添加同时包含Astra控制中心和Astra控制服务的集群。

使用主要云提供商之一(AKS、EKS、GKE）提供的Kubernetes服务时、不需要添加集群。

第1步：获取kubeconfig

您需要从Kubernetes管理员或服务处获取一份* kubconfig*文件副本。

第2步：准备kubeconfig

在使用* kubeconfig*文件之前、应执行以下操作：

1. 将文件从YAML格式转换为JSON：

   如果您收到格式为YAML的kubeconfig文件、则需要将其转换为JSON。

2. 在base64中编码JSON：

   必须在base64中对JSON文件进行编码。

示例

以下是将kubeconfigfile文件从YAML转换为JSON并在base64中进行编码的示例：

```
yq -o=json ~/.kube/config | base64
```

第3步：选择云

执行工作流 "列出云" 并选择要添加集群的云。

您只能选择*私有*云。

第4步：创建凭据

执行以下REST API调用以使用kubeconfig文件创建凭据。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/credentials

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/credentials" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-credential",
  "version" : "1.1",
  "name" : "Cloud One",
  "keyType" : "kubeconfig",
  "keyStore" : {
    "base64": encoded_kubeconfig
  },
  "valid" : "true"
}
```

第5步：添加集群

执行以下REST API调用以将集群添加到云。的值 `credentialID` 输入字段是从上一步中的REST API调用获取的。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /topology/v1/cloud / {clune_id} /集群

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/clouds/<CLOUD_ID
>/clusters" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type" :  "application/astra-cluster",
  "version" : "1.1",
  "credentialID": credential_id
}
```

列出受管集群

您可以列出当前由 Astra 管理的 Kubernetes 集群。

执行以下 REST API 调用。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /topology/v1/managedClusters

curl 示例：返回所有集群的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/managedClusters"
\
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

管理集群

您可以管理Kubernetes集群、以便执行数据保护。

第1步：选择要管理的集群

执行工作流 "列出集群" 并选择所需的集群。属性 `managedState` 的集群必须为 `unmanaged`。

第2步：(可选)选择存储类

也可以执行工作流 "列出存储类" 并选择所需的存储类。

如果在管理集群的调用中未提供存储类、则会使用默认存储类。

第3步：管理集群

执行以下REST API调用以管理集群。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /topology/v1/managedClusters

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/managedClusters"
\
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例

```
{
  "type": "application/astra-managedCluster",
  "version": "1.0",
  "id": "d0fdf455-4330-476d-bb5d-4d109714e07d"
}
```

云

列出云

您可以列出特定Astra帐户定义和可用的云。

执行以下REST API调用以列出云。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/｛account_id｝/topology/v1/Clouds

curl示例：返回所有云的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/clouds" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

存储分段

列出分段

您可以列出为特定 Astra 帐户定义的 S3 存储分段。

执行以下REST API调用以列出分段。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/｛account_id｝/topology/v1/bass桶

curl 示例：返回所有存储分段的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/buckets" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

存储

列出存储类

您可以列出可用的存储类。

第1步：选择云

执行工作流 "列出云" 并选择要使用的云。

第2步：选择集群

执行工作流 "列出集群" 并选择集群。

第3步：列出特定集群的存储类

执行以下REST API调用以列出特定集群和云的存储类。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/ {account_id} /topology/v1/clones/clones/clusters/storageClasses.<CLUSTER_ID><CLOUD_ID>

curl示例：返回所有存储类的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/clouds/<CLOUD_ID
>/clusters/<CLUSTER_ID>/storageClasses" \
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 输出示例

```
{
    "items": [
        {
            "type": "application/astra-storageClass",
            "version": "1.1",
            "id": "4bacbb3c-0727-4f58-b13c-3a2a069baf89",
            "name": "ontap-basic",
            "provisioner": "csi.trident.netapp.io",
            "available": "eligible",
            "allowVolumeExpansion": "true",
            "reclaimPolicy": "Delete",
            "volumeBindingMode": "Immediate",
```

```
            "isDefault": "true",
            "metadata": {
                "createdBy": "system",
                "creationTimestamp": "2022-10-26T05:16:19Z",
                "modificationTimestamp": "2022-10-26T05:16:19Z",
                "labels": []
            }
        },
        {
            "type": "application/astra-storageClass",
            "version": "1.1",
            "id": "150fe657-4a42-47a3-abc6-5dafba3de8bf",
            "name": "thin",
            "provisioner": "kubernetes.io/vsphere-volume",
            "available": "ineligible",
            "reclaimPolicy": "Delete",
            "volumeBindingMode": "Immediate",
            "metadata": {
                "createdBy": "system",
                "creationTimestamp": "2022-10-26T04:46:08Z",
                "modificationTimestamp": "2022-11-04T14:58:19Z",
                "labels": []
            }
        },
        {
            "type": "application/astra-storageClass",
            "version": "1.1",
            "id": "7c6a5c58-6a0d-4cb6-98a0-8202ad2de74a",
            "name": "thin-csi",
            "provisioner": "csi.vsphere.vmware.com",
            "available": "ineligible",
            "allowVolumeExpansion": "true",
            "reclaimPolicy": "Delete",
            "volumeBindingMode": "WaitForFirstConsumer",
            "metadata": {
                "createdBy": "system",
                "creationTimestamp": "2022-10-26T04:46:17Z",
                "modificationTimestamp": "2022-10-26T04:46:17Z",
                "labels": []
            }
        },
        {
            "type": "application/astra-storageClass",
            "version": "1.1",
            "id": "7010ef09-92a5-4c90-a5e5-3118e02dc9a7",
            "name": "vsim-san",
```

```
            "provisioner": "csi.trident.netapp.io",
            "available": "eligible",
            "allowVolumeExpansion": "true",
            "reclaimPolicy": "Delete",
            "volumeBindingMode": "Immediate",
            "metadata": {
                "createdBy": "system",
                "creationTimestamp": "2022-11-03T18:40:03Z",
                "modificationTimestamp": "2022-11-03T18:40:03Z",
                "labels": []
            }
        }
    ]
}
```

列出存储后端

您可以列出可用的存储后端。

执行以下 REST API 调用。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
获取	/accouns/｛account_id｝/topology/v1/storageBackend

curl 示例：返回所有存储后端的所有数据

```
curl --request GET \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/topology/v1/storageBackends"
\
--include \
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN"
```

JSON 输出示例


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


为自管理集群启用动态ANF池

在具有ANF存储后端的私有内部集群中备份托管应用程序时、您必须启用动态ANF池功能。为此、可以提供一个订阅ID、以便在扩展和缩减容量池时使用。

> 动态ANF池是使用Azure NetApp Files (ANF)存储后端的Astra托管应用程序的一项功能。备份这些应用程序时、A作用 力会自动扩展和收缩永久性卷所属的容量池1.5倍。这样可以确保有足够的空间进行备份、而不会产生额外的永久费用。请参见 "Azure应用程序备份" 有关详细信息 …

第**1**步：添加**Azure**订阅标识符

执行以下 REST API 调用。

您需要根据环境的具体情况更新JSON输入示例、包括订阅ID和服务主体的base64值。

HTTP方法和端点

此REST API调用使用以下方法和端点。

HTTP 方法	路径
发布	/accouns/ {account_id} /core/v1/credentials

curl 示例

```
curl --request POST \
--location
"https://astra.netapp.io/accounts/$ACCOUNT_ID/core/v1/credentials" \
--include \
--header "Content-Type: application/astra-credential+json"
--header "Accept: */*" \
--header "Authorization: Bearer $API_TOKEN" \
--data  @JSONinput
```

JSON 输入示例


```
{
  "keyStore": {
    "privKey": "SGkh",
    "pubKey": "UGhpcyCpcyBhbiBleGFtcGxlLg==",
    "base64":
"fwogICAgJmFwcElkIjogIjY4ZmSiODFiLTY0YWYtNDdjNC04ZjUzLWE2NDdlZTUzMGZkZCIsC
iAgICAiZGlzcGxheU5hbWUiOiAic3AtYXN0cmEtZGV2LXFhIiwKICAgICJuYW1lIjogImh0dHA
6Ly9zcC1hc3RyYS1kZXYtcWEiLAogICAgInBhc3N3b3JkIjogIllLQThRfk9IVVJkZWZYM0pST
WJlLnpUeFBleVE0UnNwTG9DcUJjazAiLAogICAgInRlbmFudCI6ICIwMTFjZGY2Yy03NTEyLTQ
3MDUtYjI0ZS03NzIxYWZkOGNhMzciLAogICAgInN1YnNjcmlwdGlvbklkIjogImIyMDAxNTVmL
TAwMWEtNDNiZS04N2JlLTNlZGRlODNhY2VmNCIKfQ=="
  },
  "name": "myCert",
  "type": "application/astra-credential",
  "version": "1.1",
  "metadata": {
    "labels": [
      {
        "name": "astra.netapp.io/labels/read-only/credType",
        "value": "service-account"
      },
      {
        "name": "astra.netapp.io/labels/read-only/cloudName",
        "value": "OCP"
      },
      {
        "name": "astra.netapp.io/labels/read-only/azure/subscriptionID",
        "value": "b212156f-001a-43be-87be-3edde83acef5"
      }
    ]
  }
}
```


第**2**步：根据需要添加存储分段

如果需要、您应向受管应用程序添加存储分段。

第**3**步：备份托管应用程序

执行工作流 "为应用程序创建备份"。存在初始永久性卷的容量池将自动扩展和缩减。

第**4**步：查看事件日志

备份期间会记录活动事件。执行工作流 "列出通知" 可查看信息。