Infrastructure workflows

Astra Automation

NetApp
August 11, 2025

Table of Contents

Infrastructure workflows

Prepare to use the infrastructure workflows

You can use these workflows to create and maintain the infrastructure used with an Astra Control Center deployment. In many cases, the workflows can also be used with Astra Control Service.

These workflows can be expanded and enhanced by NetApp at any time and so you should review them periodically.

General preparation

Before using any of the Astra workflows, make sure to review Prepare to use the workflows.

Workflow categories

The infrastructure workflows are organized in different categories to make it easier to locate the one you want.

Category	Description
Identity and access	These workflows allow you to manage identity and how Astra is accessed. The resources include users, credentials, and tokens.
LDAP configuration	You can optionally configure Astra Control Center to use LDAP to authenticate selected users.
Clusters	You can add managed Kubernetes clusters which allows you to protect and support the applications they contain.
Clouds	These workflows provide access to the clouds available through the Astra Control REST API.
Buckets	You can use these workflows to create and manage the S3 buckets used to store backups.
Storage	These workflows allow you to add and maintain storage backends and volumes.

Identity and access

List the users

You can list the users that are defined for a specific Astra account.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/core/v1/users

Additional input parameters

In addition to the parameters common with all REST API calls, the following parameters are also used in the curl examples for this step.

Parameter	Type	Required	Description
include	Query	No	Optionally select the values you want returned in the response.

Curl example: Return all data for all users

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

Curl example: Return the first name, last name, and id for all user

```
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 output example

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

Create a user

You can create a user with specific credentials and a pre-defined role. You can also optionally restrict the user's access to specific namespaces.

Step 1: Select a user name

Perform the workflow List users and select an available name not currently in use.

Step 2: Create the user

Perform the following REST API call to create a user. After successful completion of the call, the new user will not yet be usable.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/users

Curl example

```
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 input example

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

JSON output example


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


Step 3: Optionally select the allowed namespaces

Perform the workflow List the namespaces and select the namespaces you want to restrict access to.

Step 4: Bind the user to a role

Perform the following REST API call to bind the user to a role. The example below places no restrictions on the namespace access. See Enhanced RBAC with namespace granularity for more information.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/roleBindings

Curl example

```
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 input example

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

Step 5: Create a credential

Perform the following REST API call to create a credential and associate it with the user. This example uses a password which is provided as a base64 value. The `name` property should contain the ID of the user returned in the previous step. The input property `change` must also be encoded in base64 and determines if the user must change their password at first login (`true` or `false`).

> This step is only required with Astra Control Center deployments using local authentication. It is not needed with Astra Control Center deployments using LDAP or with Astra Control Service deployments.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/credentials

Curl example

```
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 input example

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

LDAP configuration

Prepare for LDAP configuration

You can optionally integrate Astra Control Center with a Lightweight Directory Access Protocol (LDAP) server to perform authentication for selected Astra users. LDAP is an industry standard protocol for accessing distributed directory information and a popular choice for enterprise authentication.

Related information

- LDAP Technical Specification Road Map
- LDAP version 3

Overview of the implementation process

At a high level, there are several steps you need to perform to configure an LDAP server to provide authentication for Astra users.

> While the steps presented below are in a sequence, in some cases you can perform them in a different order. For example, you can define the Astra users and groups before configuring the LDAP server.

1. Review Requirements and limitations to understand the options, requirements, and limitations.
2. Select an LDAP server and the desired configuration options (including security).
3. Perform the workflow Configure Astra to use an LDAP server to integrate Astra with the LDAP server.
4. Review the users and groups at the LDAP server to make sure they are defined properly.
5. Perform the appropriate workflow in Add LDAP entries to Astra to identify the users to be authenticated using LDAP.

Requirements and limitations

You should review the Astra configuration essentials presented below, including limitations and configuration options, before configuring Astra to use LDAP for authentication.

Only supported with Astra Control Center

The Astra Control platform provides two deployment models. LDAP authentication is only supported with Astra Control Center deployments.

Configuration using REST API or web user interface

The current release of Astra Control Center supports configuration of LDAP authentication using both the Astra Control REST API as well as the Astra web user interface.

LDAP server required

You must have an LDAP server to accept and process the Astra authentication requests. Microsoft's Active Directory is supported with the current Astra Control Center release.

Secure connection to the LDAP server

When configuring the LDAP server in Astra, you can optionally define a secure connection. In this case a certificate is needed for the LDAPS protocol.

Configure users or groups

You need to select the users to be authenticated using LDAP. You can do this either by identifying the individual users or a group of users. The accounts must be defined at the LDAP server. They also need to be identified in Astra (type LDAP) which allows the authentication requests to be forwarded to LDAP.

Role constraint when binding a user or group

With the current release of Astra Control Center, the only supported value for `roleConstraint` is "*". This indicates the user is not restricted to a limited set of namespaces and can access all of them. See Add LDAP entries to Astra for more information.

LDAP credentials

The credentials used by LDAP include the username (email address) and the associated password.

Unique email addresses

All email addresses acting as usernames in an Astra Control Center deployment must be unique. You cannot add an LDAP user with an email address that is already defined to Astra. If a duplicate email exists, you need to first delete it from Astra. See Remove users at the Astra Control Center documentation site for more information.

Optionally define LDAP users and groups first

You can add the LDAP users and groups to Astra Control Center even if they don't yet exist in LDAP or if the LDAP server is not configured. This allows you to preconfigure the users and groups before configuring the LDAP server.

A user defined in multiple LDAP groups

If an LDAP user belongs to multiple LDAP groups and the groups have been assigned different roles in Astra, the user's effective role when authenticating will be the most privileged. For example, if a user is assigned the `viewer` role with group1 but has the `member` role in group2, the user's role would be `member`. This is based on the hierarchy used by Astra (highest to lowest):

- Owner
- Admin
- Member
- Viewer

Periodic account synchronization

Astra synchronizes it's users and groups with the LDAP server approximately every 60 second. So if a user or group is added to or removed from LDAP, it can take up to one minute before it is available in Astra.

Disabling and resetting the LDAP configuration

Before attempting to reset the LDAP configuration, you must first disable LDAP authentication. Also, to change the LDAP server (`connectionHost`), you need to perform both operations. See Disable and reset LDAP for more information.

REST API parameters

The LDAP configuration workflows make REST API calls to accomplish the specific tasks. Each API call can include input parameters as shown in the provided samples. See Online API reference for information about how to locate the reference documentation.

Configure Astra to use an LDAP server

You need to select an LDAP server and configure Astra to use the server as an authentication provider. The configuration task consists of the steps described below. Each step includes a single REST API call.

Step 1: Add a CA certificate

Perform the following REST API call to add a CA certificate to Astra.

This step is optional and only required if you want Astra and the LDAP to communicate over a secure channel using LDAPS.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/certificates

Curl example

```
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 input example

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

Note the following about the input parameters:

- `cert` is a JSON string containing a base64 encoded PKCS-11 formatted certificate (PEM encoded).
- `isSelfSigned` should be set to `true` if the certificate is self-signed. The default is `false`.

JSON output example


```
{
  "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": []
    }
}
```

Step 2: Add the bind credentials

Perform the following REST API call to add the bind credentials.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/credentials

Curl example

```
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 input example

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

Note the following about the input parameters:

- `bindDn` and `password` are the base64 encoded bind credentials of the LDAP admin user that is able to connect and search the LDAP directory. `bindDn` is the LDAP user's email address.

JSON output example

```
{
    "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"
    }
}
```

Note the following the response parameters:

- The `id` of the credential is used in subsequent workflow steps.

Step 3: Retrieve the UUID of the LDAP setting

Perform the following REST API call to retrieve the UUID of the `astra.account.ldap` setting that is included with Astra Control Center.

> The curl example below uses a query parameter to filter the settings collection. You can instead remove the filter to get all the settings and then search for `astra.account.ldap`.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/core/v1/settings

Curl example

```
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 output example

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

Step 4: Update the LDAP setting

Perform the following REST API call to update the LDAP setting and complete the configuration. Use the `id` value from the previous API call for the `<SETTING_ID>` value in the URL path below.

You can issue a GET request for the specific setting first to see the configSchema. This will provide more information about the required fields in the configuration.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PUT	/accounts/{account_id}/core/v1/settings/{setting_id}

Curl example

```
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 input example

```
{
  "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"
    }
}
```

Note the following about the input parameters:

- `isEnabled` should be set to `true` or an error may occur.
- `credentialId` is the id of the bind credential created earlier.
- `secureMode` should be set to `LDAP` or `LDAPS` based on your configuration in the earlier step.
- Only 'Active Directory' is supported as a vendor.

If the call is successful, the HTTP 204 response is returned.

Step 5: Retrieve the LDAP setting

You can optionally perform the following REST API call to retrieve the LDAP settings and confirm the update.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/core/v1/settings/{setting_id}

Curl example

```
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 output example


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

Locate the `state` field in the response which will have one of the values in the table below.

State	Description
pending	The configuration process is still active and not completed yet.
valid	Configuration has been completed successfully and `currentConfig` in the response matches `desiredConfig`.
error	The LDAP configuration process failed.

Add LDAP entries to Astra

After LDAP is configured as an authentication provider for Astra Control Center, you can select the LDAP users that Astra will authenticate using the LDAP credentials. Each user must have a role in Astra before they can access Astra through the Astra Control REST API.

There are two ways you can configure Astra to assign roles. Choose the one that is appropriate for your environment.

- Add and bind an individual user
- Add and bind a group

> The LDAP credentials are in the form of a username as an email address and the associated LDAP password.

Add and bind an individual user

You can assign a role to each Astra user which is used after LDAP authentication. This is appropriate when there is a small number of users and each might have different administrative characteristics.

Step 1: Add a user

Perform the following REST API call to add a user to Astra and indicate that LDAP is the authentication provider.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/users

Curl example

```
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 input example

```
{
  "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"
}
```

Note the following about the input parameters:

- The following parameters are required:
  - `authProvider`
  - `authID`
  - `email`
- `authID` is the distinguished name (DN) of the user in LDAP
- `email` must be unique for all users defined in Astra

If the `email` value is not unique, an error occurs and a 409 HTTP status code is returned in the response.

JSON output example


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


Step 2: Add a role binding for the user

Perform the following REST API call to bind the user to a specific role. You need to have the UUID of the user created in the previous step.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/roleBindings

Curl example

```
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 input example

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

Note the following about the input parameters:

- The value used above for `roleConstraint` is the only option available for the current release of Astra. It indicates the user is not restricted to a limited set of namespaces and can access them all.

JSON response example

```
{
  "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": ["*"]
}
```

Note the following about the response parameters:

- The value `user` for the `principalType` field indicates the role binding was added for a user (not a group).

Add and bind a group

You can assign a role to an Astra group which is used after LDAP authentication. This is appropriate when there is a large number of users and each might have similar administrative characteristics.

Step 1: Add a group

Perform the following REST API call to add a group to Astra and indicate that LDAP is the authentication provider.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/groups

Curl example

```
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 input example

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

Note the following about the input parameters:

- The following parameters are required:
  - `authProvider`
  - `authID`

JSON response example

```
{
  "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": []
  }
}
```

Step 2: Add a role binding for the group

Perform the following REST API call to bind the group to a specific role. You need to have the UUID of the group created in the previous step. Users that are members of the group will be able to sign in to Astra after LDAP performs the authentication.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/roleBindings

Curl example

```
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 input example

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

Note the following about the input parameters:

- The value used above for `roleConstraint` is the only option available for the current release of Astra. It indicates the user is not restricted to certain namespaces and can access them all.

JSON response example

```
{
  "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": ["*"]
}
```

Note the following about the response parameters:

- The value `group` for the `principalType` field indicates the role binding was added for a group (not a user).

Disable and reset LDAP

There are two optional though related administrative tasks you can perform as needed for an Astra Control Center deployment. You can globally disable LDAP authentication and reset the LDAP configuration.

Both workflow tasks require the id for the `astra.account.ldap` Astra setting. Details for how to retrieve the setting id are included in **Configure the LDAP server**. See Retrieve the UUID of the LDAP setting for more information.

- Disable LDAP authentication
- Reset the LDAP authentication configuration

Disable LDAP authentication

You can perform the following REST API call to globally disable LDAP authentication for a specific Astra deployment. The call updates the `astra.account.ldap` setting and the `isEnabled` value is set to `false`.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PUT	/accounts/{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 input example

```
{
  "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"
    }
}
```

If the call is successful, the `HTTP 204` response is returned. You can optionally retrieve the configuration settings again to confirm the change.

Reset the LDAP authentication configuration

You can perform the following REST API call to disconnect Astra from the LDAP server and reset the LDAP configuration in Astra. The call updates the `astra.account.ldap` setting and the value of `connectionHost` is cleared.

The value of `isEnabled` must also be set to `false`. You can either set this value before making the reset call or as part of making the reset call. In the second case, `connectionHost` should be cleared and `isEnabled` set to false on the same reset call.

This is a disruptive operation and you should proceed with caution. It deletes all the imported LDAP users and groups. It also deletes all the related Astra users, groups, and roleBindings (LDAP type) that you created in Astra Control Center.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
PUT	/accounts/{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 input example

```
{
  "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"
    }
}
```

Note the following:

- To change the LDAP server, you must both disable and reset LDAP changing `connectHost` to a null value as shown in the example above.
- If the call is successful, the `HTTP 204` response is returned. You can optionally retrieve the configuration again to confirm the change.

Clusters

List the clusters

You can list the available clusters in a specific cloud.

Step 1: Select the cloud

Perform the workflow List the clouds and select the cloud containing the clusters.

Step 2: List the clusters

Perform the following REST API call to list the clusters in a specific cloud.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/clouds/{cloud_id}/clusters

Curl example: Return all data for all clusters

```
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 output example


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

Add a cluster using credentials

You can add a cluster so it will be available to be managed by Astra. Beginning with the Astra 22.11 release, you can add a cluster with both Astra Control Center and Astra Control Service.

Adding a cluster is not required when using a Kubernetes service from one of the major cloud providers (AKS, EKS, GKE).

Step 1: Obtain the kubeconfig file

You need to obtain a copy of the **kubconfig** file from your Kubernetes administrator or service.

Step 2: Prepare the kubeconfig file

Before using the **kubeconfig** file, you should perform the following operations:

1. Convert file from YAML format to JSON:

   If you receive the kubeconfig file formatted as YAML, you need to convert it to JSON.

2. Encode JSON in base64:

   You must encode the JSON file in base64.

Example

Here is an example of converting the kubeconfig file from YAML to JSON and then encoding it in base64:

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

Step 3: Select the cloud

Perform the workflow List the clouds and select the cloud where the cluster will be added.

The only cloud you can select is the **private** cloud.

Step 4: Create a credential

Perform the following REST API call to create a credential using the kubeconfig file.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/credentials

Curl example

```
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 input example

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

Step 5: Add the cluster

Perform the following REST API call to add the cluster to the cloud. The value of the `credentialID` input field is obtained from the REST API call in the previous step.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/topology/v1/clouds/{cloud_id}/clusters

Curl example

```
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 input example

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

List managed clusters

You can list the Kubernetes clusters currently managed by Astra.

Perform the following REST API call.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/managedClusters

Curl example: Return all data for all clusters

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

Manage a cluster

You can manage a Kubernetes cluster so that data protection can be performed.

Step 1: Select the cluster to manage

Perform the workflow List clusters and select the desired cluster. The property `managedState` of the cluster must be `unmanaged`.

Step 2: Optionally, select the storage class

Optionally perform the workflow List storage classes and select the desired storage class.

If you don't provide a storage class on the call to manage the cluster, your default storage class will be used.

Step 3: Manage the cluster

Perform the following REST API call to manage the cluster.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/topology/v1/managedClusters

Curl example

```
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 input example

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

Clouds

List the clouds

You can list the clouds defined and available a specific Astra account.

Perform the following REST API call to list the clouds.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/clouds

Curl example: Return all data for all clouds

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

Buckets

List the buckets

You can list the S3 buckets defined for a specific Astra account.

Perform the following REST API call to list the buckets.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/buckets

Curl example: Return all data for all buckets

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

Storage

List storage classes

You can list the available storage classes.

Step 1: Select the cloud

Perform the workflow List the clouds and select the cloud you'll be working in.

Step 2: Select the cluster

Perform the workflow List the clusters and select the cluster.

Step 3: List the storage classes for a specific cluster

Perform the following REST API call to list the storage classes for a specific cluster and cloud.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/clouds/<CLOUD_ID>/clusters/<CLUSTER_ID>/storageClasses

Curl example: Return all data for all storage classes

```
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 output example

```
{
    "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": []
            }
        }
```

```
    ]
}
```

List storage backends

You can list the available storage backends.

Perform the following REST API call.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
GET	/accounts/{account_id}/topology/v1/storageBackends

Curl example: Return all data for all storage backends

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

JSON output example


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


Enable dynamic ANF pools for self-managed clusters

When backing up a managed app in a private on-premises cluster that has an ANF storage backend, you must enable the dynamic ANF pools feature. This is done by providing a subscription ID to use when expanding and contracting the capacity pools.

Dynamic ANF pools is feature of the Astra managed apps that use an Azure NetApp Files (ANF) storage backend. When backing up these apps, Astra automatically expands and contracts the capacity pools the persistent volumes belong to by a factor of 1.5. This ensures there is enough space for the backup without incurring an additional permanent charge. See Azure application backups for more information.

Step 1: Add the Azure subscription identifier

Perform the following REST API call.

You need to update the JSON input example as appropriate for your environment, including the subscription ID and the base64 value for the service principal.

HTTP method and endpoint

This REST API call uses the following method and endpoint.

HTTP method	Path
POST	/accounts/{account_id}/core/v1/credentials

Curl example

```
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 input example


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


Step 2: Add a bucket if needed

You should add a bucket to the managed application if needed.

Step 3: Take a backup of the managed app

Perform the workflow Create a backup for an app. The capacity pool where the original persistent volume is present will expand and shrink automatically.

Step 4: Review the event log

Activity events are logged during the backup. Perform the workflow List the notifications to view the messages.

Copyright information

Trademark information